



cerenade

Visual eForms Designer
Visual eForms ToolBox
User Manual

Contents

Introduction	1
How to Use This Manual	1
Conventions	2
Help and Support	2
Installation and Setup	3
Screen Components	6
Menus	6
Toolbars	6
Object Toolbar	6
Standard Toolbar	9
.....	11
Properties	11
Properties Window	11
Alternative Properties Window	11
Properties Control Bar	12
Status Bar	12
Form Design Guide	14
Design Mode versus Fill Mode	14
Working with Forms	15
Opening Forms	15
New Forms	16
Defining Form Setup	17
Defining Page Setup	18
Saving Forms	19
Setting Form Options	20
Setting Form Zoom	21
Adding and Removing Pages	22
GoTo Page	22
Setting Form Properties	24
Import/Export	26
Working With Objects	28
Non-Fillable Objects	28
Fillable Objects	29
Drawing Objects	30
Crosshairs	31
Positioning Objects	32
Selecting Objects	32
Ruler	34
Grid	34
Using Snap to Grid	36
Aligning Objects	37
Modifying Objects	38
Changing Object Properties	39
Using the Properties Windows	39
Miscellaneous	40
.....	40
Setting Default Object Properties	40
Position	42
.....	43
Appearance	43
Borders	45

Margins	47
Text	48
Mixed Fonts	51
RTL	51
Edit	52
Notify	57
Special Properties.....	59
Masks	59
Example of a Mask	60
Check Boxes	62
Radio Buttons	63
Buttons	64
Images	65
Editable Images	66
Drop Lists	67
Tables	69
Bar Codes	73
Digital Signatures	75
Alternative Properties Window	79
Property Control Bar	80
Editing Text in Forms	83
Check Spelling	84
Find and Replace	85
Printing Forms	86
Filling Forms	87
Sticky Notes	88
Type Anywhere	88
Accessibility	90
Making Forms Accessible	91
Scripts	94
Details of Scripts	96
Functions.....	96
Built-in Functions	98
Expressions	101
Operators.....	102
Creating the script	103
Built-In Functions	105
ALERT	105
CLEARDATA	106
DATE	107
DAY	108
DIFFDATE	109
DIFFTIME	110
GETCURRFIELD	111
GETCURRPAGE	112
GETFIELDHELP	113
GETFIELDPROPERTY	114
GETNUMPAGES	116
GETUNFILLEDMANDATORY	117
GOTOFIELD	118
GOTOPAGE	119
HOUR	120
IF	121
LEFT	122

LOWER	123
LTRIM	124
MINUTE	125
MONTH	126
NUM	127
PRINTDIALOG	128
RIGHT	129
ROUND	130
RTRIM	131
SEC	132
SETFIELDDBDATA	133
SETFIELDPROPERTY	134
SETPROPERTY	135
STR	136
STRAT	137
STREXTRACT	138
STRINSTR	139
STRLEN	140
SUM	141
SUMDATE	142
SUMTIME	143
TIME	144
UPPER	145
YEAR	146
Databases	147
Database Connectivity	147
Coding	147
MmaADOi	148
Choosing a Database Format	149
Database Relations	150
Database Relations Screen	150
Toolbar	151
Legend	151
Primary and Secondary Data Source Windows	151
Unassigned Column	151
Primary and Secondary Data Sources	152
Assigning Database Relations	154
Using Database Relations	158
Example of code in Visual Basic	158
Advanced Programming	159
Filler Active X	160
Properties	160
DefaultPath	160
FormName	161
FormVersion	162
LastErrorCode	163
LastErrorDesc	165
vfBackColor	166
vfBottomBorder	167
vfButton	168
vfCheckBox	169
vfDate	170
vfEditableImage	171
vfEnabled	172

vfFillableText	173
vfLeftBorder	174
vfLineColor	175
vfMaxFillChars	176
vfNumber	177
vfPageNumber	178
vfRightBorder	179
vfRoundedBorder	180
vfSignature	181
vfTextColor	182
vfTopBorder	183
vfType	184
vfVisible	185
ZoomFactor	186
Methods.....	187
AbandonChanges	187
AboutBox	188
AppendField	189
AutoReduceFonts	190
ClearData	191
CloseForm	192
Copy	193
CreateNote	194
Cut	196
DisableRedraw	197
DropListAddString	198
DropListClear	200
DropListDeleteString	201
DropListGetCount	202
DropListGetCurSel	203
DropListSetCurSel	204
EnableAddendumTag	205
EnableField	206
EnableFieldAddendumTag	207
EnableFields	208
FileDialog	209
FillTestData	211
GetCurrField	212
GetCurrPage	213
GetFieldAddendumLen	214
GetFieldAddendumText	215
GetFieldCount	219
GetFieldHelp	220
GetFieldLen	221
GetFieldLineCount	222
GetFieldLong	223
GetFieldProperty	224
GetFieldString	233
GetFieldTextWidth	234
GetFirstField	235
GetFormPath	239
GetFormProperty	240
GetFormWindow	242
GetNextField	243

GetNumPages	244
GetSignatureTimestamp	245
GetSignerName	246
GetUnfilledMandatory	247
GetVersion	248
GotoField	249
GotoFieldByTabOrder	250
GotoPage	251
HighlightFields	252
ImportAscii	253
IsFormChanged	254
IsFormLocked	255
LockForm	256
MAPISendMail	257
NextField	258
OnPrintText	259
OpenForm	261
OpenFormData	262
OpenFormDataDialog	264
OpenFormDialog	265
OpenFormPasswordDecrypt	266
OpenInternetForm	267
Paste	268
PrevField	269
Print	270
PrintAbort	271
PrintAddendum	272
PrintDialog	273
PrintEnd	274
PrintForm	275
PrintFreeDC	276
PrintGetDC	277
PrintGetParams	278
PrintPage	279
PrintStart	280
Redraw	281
SaveForm	282
SaveFormData	283
SaveFormDataDialog	284
SaveFormDialog	285
SaveFormPasswordEncrypt	286
Scroll	287
SetCursorPosition	288
SetFieldData	289
SetEnterpriseParams	290
SetFieldDataEx	291
SetFieldProperty	292
SetFieldSize	294
SetFocus	295
SetFormProperty	296
SetNotifyOnCalc	298
SetSharedFontTable	299
ShowNonPrintables	300
SignForm	301

Undo	303
UnsignForm	304
ValidateSignature	306
VarGetCurrField	307
VarGetFieldAddendumText	308
VarGetFieldHelp	309
VarGetFieldString	310
VarOpenFormData	311
VarOpenFormDataDialog	313
ViewEnlarge	314
ViewFitSides	315
ViewRealSize	316
XMLGetFormData	317
XMLSetFormData	320
Events	323
FieldClick	323
FieldDbClick	324
FieldModified	325
FieldGotFocus	326
FieldLostFocus	327
FieldMouseEnter	328
FieldMouseExit	329
FillerLoaded	330
GotFocus	331
LostFocus	332
OnChar	333
OnError	335
PageChange	336
Database ActiveX	337
Properties	337
Caption Property	337
Methods	338
AboutBox	338
AddNew	339
Connect	340
CreateDatabase	341
Delete	342
Disconnect	343
FindFirst FindLast FindNext FindPrevious	344
GetAbsolutePosition	345
GetLastError	346
GetRecordCount	347
Lookup	348
MoveFirst MoveLast MoveNext MovePrevious	349
Update	350
Events	351
RecordAdd	351
RecordDelete	352
RecordMove	353
RecordUpdate	354
Index	355

Introduction

Using This Manual

This Installation Manual contains procedures for installing the Visual eForms Enterprise Server. This manual accompanies the Visual eForms Enterprise Server User Manual, Admin Manual, and Specifications Manual. The Visual eForms Designer and ToolBox User Manual provides detailed information on creating forms for use with the Enterprise Server.

This preface provides an overview of the Visual eForms Enterprise Server Installation Guide, explaining conventions used in this manual.

Conventions

As you work with Visual eForms Enterprise Server and its manual, remember that the text format indicates a specific action or meaning. The following table lists the conventions used in this manual.

Convention	Meaning
Bold	In procedures, indicates text that you type or the name of screen objects such as icons or buttons.
>Bold	Identifies a procedure.
SMALL CAPS	Refers to keys, such as SHIFT, CONTROL, or TAB.
“Quotation marks”	Web links and folder names.

System Requirements

Client PC:

For the web application to function properly, the following must be installed on each client PC:

- **IE.** Supported Versions of Windows and IE:
 - Windows 95, 98, Me, NT, 2000, and IE 4.0 or higher
- **Netscape.** Supported Versions of Windows and Netscape:
 - Windows 95, 98, Me, NT, 2000, and Netscape 4.X, 5.X (Netscape 6.X is not yet supported).
- **Mac, Unix.** The client PC can be a Non-Windows PC. Forms are presented in HTML to this group of PCs.

Minimum Server Requirements:

Pentium® II 300 MHz, with 256 MB RAM, 10 MB available hard disk, and one of the following operating systems:

- Microsoft Windows 2000
- NT Server 4.0 SP 4 and above
- Solaris 2.6, 7, & 8
- Variety of Linux distributions
- AIX 4.3.3 or HP-UX 11.0

Database Requirements:

Microsoft SQL Server 6.5, 7.0, or 2000, Oracle 7.3 or 8.0

Messaging:

Microsoft Exchange 5.x, Lotus Notes 4.5 and 4.6, Novel GroupWise

Directory Services:

Active directory, LDAP server (Exchange 5.5 or iPlanet directory server)

Application Development:

Windows 9x, 2000, Me, or Windows NT

Web Server: Microsoft IIS 4.0, 5.0, iPlanet + Chili!Soft add-on.

Minimum Client Requirements:

Pentium 75 MHz, with 64MB RAM, 1 MB available hard disk for WYSIWYG forms, and no hard disk requirements for HTML forms.

Supported Forms Formats:

HTML and Visual eForms file format. In addition, FormFlow 1.x, 2.x, OmniForm 4.0 file formats are supported after an import to Visual eForms.

Form Design Guide

This chapter covers the basics of creating forms with Visual eForms Designer. It discusses the following topics:

- Opening, saving, and renaming existing forms.
- Creating new forms and defining their setup.
- Drawing fillable and non-fillable objects.
- Changing the appearance of objects.
- Changing the properties of objects.
- Previewing forms.

Design Mode versus Preview Mode versus Filler

Visual eForms Designer has two modes: **Design** mode and **Preview** mode. This chapter explains how to create forms and objects in **Design** mode. After you have created your form, you may want to view the form in **Preview** mode. **Preview** mode allows the form designer to see what the document will look like to the user. **Preview** mode is explained at the end of this chapter (“Previewing Forms” on page 89).

The term “**Filler**” used in this document refers to the end user completing or filling out the form using the Filler application.

Working with Forms

Opening Forms

You can open and modify existing forms in the Visual eForms Designer.

>To open an existing form

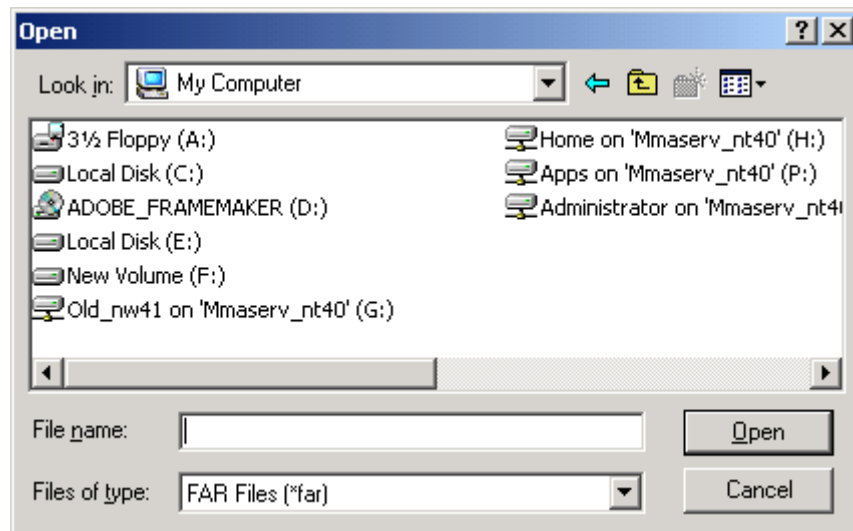
Quick key: [CTRL] +
[O].

On the **File** menu, select **Open**

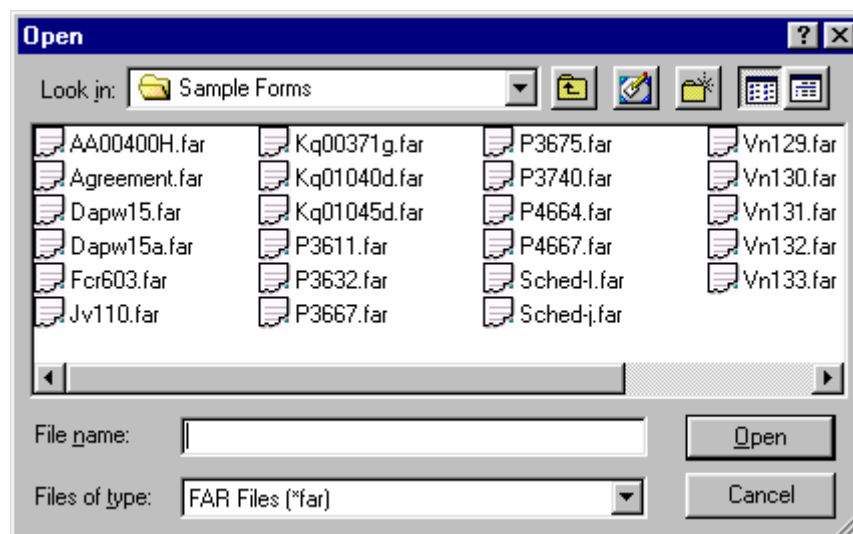
-or-

Click the **Open** icon on the **Standard** toolbar.

1. Specify the drive and directory location of the form.



2. Locate the file that you want to open and select it in the **Open** dialog box.



The file name appears in the **File name** field.

3. Click the **Open** icon.

The form appears in the designer window.

New Forms

A form is an arrangement of objects on a page. Designing a new form involves defining the following:

- Form setup - Size, Margins, and Orientation of the form.
- Drawing the outline of the objects - Box, Line, or Button.
- Defining the object's appearance - Borders or no Borders, Font size.

>To create a new form

Quick key: [CTRL] +
[N].

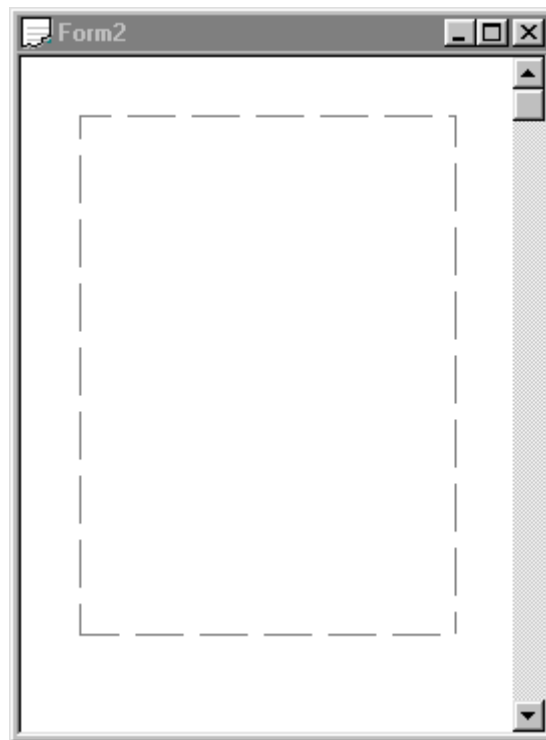
On the **File** menu, click **New**.

-or-



Click the **New** icon on the **Standard** toolbar.

A blank form window appears to begin designing your form.



Defining Form Setup

Before creating a new form you should plan the **Form Setup**. You might want to make a rough sketch on paper. Some form components may include the following:

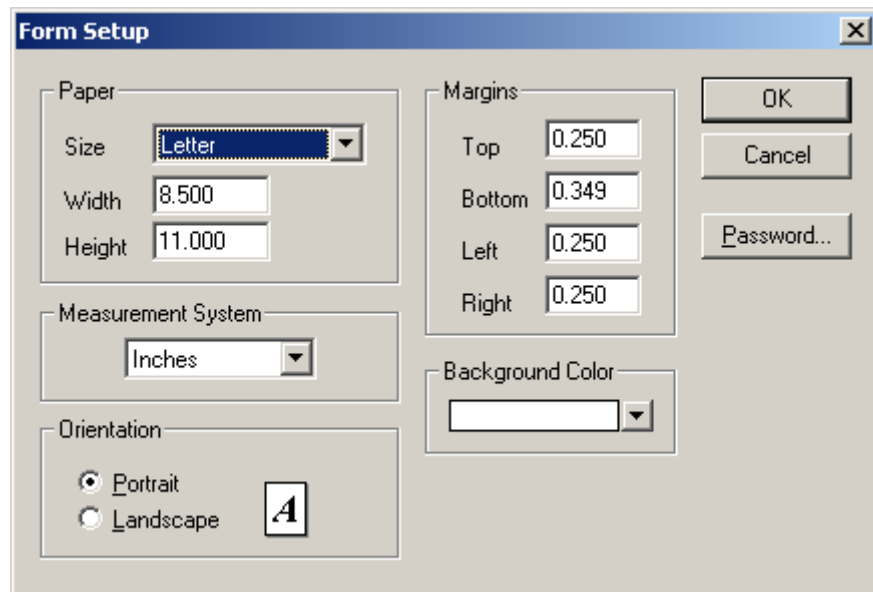
- Company name - Corporate logo and address
- Form Identification - Descriptive title and reference number
- Filling Instructions - Section headings for fields

Once you have determined the information that you would like to include in your form, you can define the layout, or **Form Setup**.

>To define the form setup

1. On the **File** menu, click **Form Setup**.

The **Form Setup** dialog box appears.

The image shows the 'Form Setup' dialog box with a blue title bar and a close button. It is divided into several sections: 'Paper' with 'Size' (set to 'Letter'), 'Width' (8.500), and 'Height' (11.000); 'Measurement System' set to 'Inches'; 'Orientation' with 'Portrait' selected and a preview icon; 'Margins' with 'Top' (0.250), 'Bottom' (0.349), 'Left' (0.250), and 'Right' (0.250); and 'Background Color' with a color selection box. On the right side, there are three buttons: 'OK', 'Cancel', and 'Password...'.

Section	Property	Value
Paper	Size	Letter
	Width	8.500
	Height	11.000
Measurement System	System	Inches
	Orientation	Portrait
Margins	Top	0.250
	Bottom	0.349
	Left	0.250
	Right	0.250
Background Color	Color	[Color Selection Box]

2. In the **Paper** section, click the **Size** drop-down menu and scroll through the list of paper sizes; then do either of the following:
 - Select one of the standard sizes.
 - Select **Custom** and specify the **Width** and **Height** of your form.
3. In the **Measurement System** section, click the drop-down menu to select inches, centimeters, or picas.
4. In the **Orientation** section, click either the **Portrait** or **Landscape** radio button.
5. In the **Margins** section, enter the width of page margins for **Top**, **Bottom**, **Left**, and **Right**. The margins appear as a dotted line around the edge of the form window.

6. On the drop-down menu, select a **Background Color** for your form.

This applies to every page in the form that you create.

7. If you wish, password protect your form by clicking **Password**, then entering and confirming a password



The password is case sensitive, so check your Caps Lock Key.

8. Click the **OK** button.

Defining Page Setup

Page setup give you the ability to customize forms on each individual page of a multi-page form. Using this new feature, you can have mixed page orientations and page sizes within the same form.

>To Define Page Setup

1. Go to the form page you wish to customize.
2. On the **File** menu, click **Page Setup**.

The Page Setup dialog box appears.

3. Make the necessary choices and click the **OK** button.

The choices you make apply to the current form page only.

Saving Forms

A new or existing form can be saved by doing the following:

>To save a form

Quick key: [CTRL] +
[S]

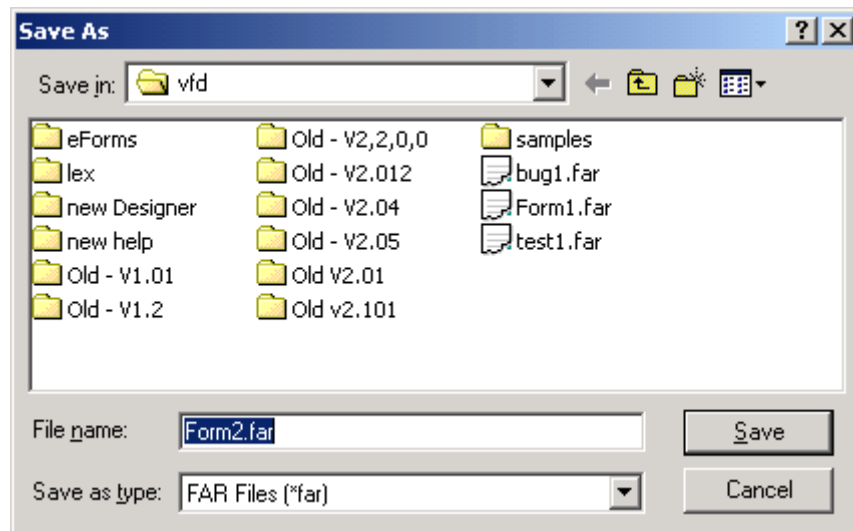
1. On the **File** menu, click **Save**.

-or-



2. Click the **Save** icon on the **Standard** toolbar.

If you are saving a new form, the **Save As** dialog box appears.



3. Enter the file name as you would like it to appear in the file directory.
4. Click the **Save** icon.

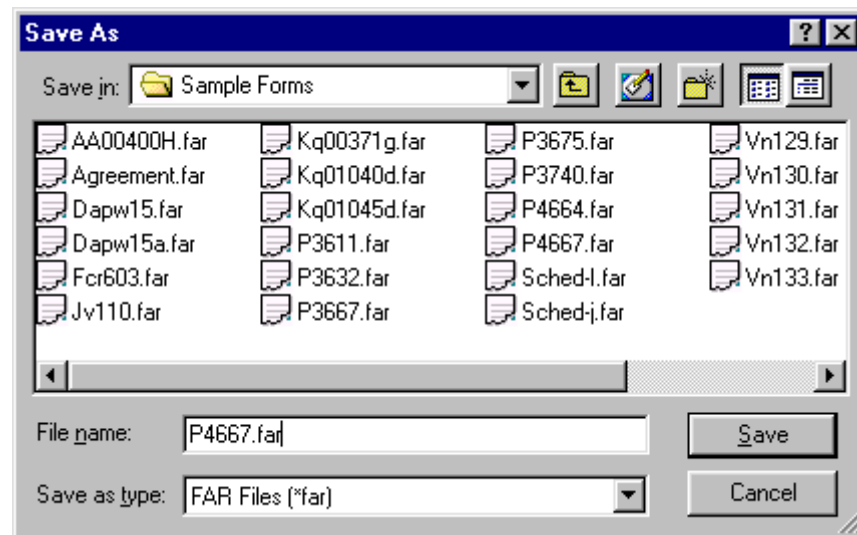
>To save an existing form under a different name

1. On the File menu, click Save As.

The **Save As** dialog box appears.



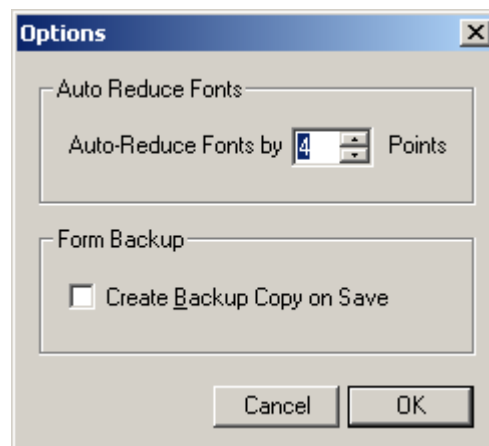
Make sure
"FAR Files
(*.far)" is
selected in
the Save as
type drop-down list.



2. Enter the file name as you would like it to appear in the file directory.
3. Click the **Save** button.

Setting Form Options

The Options dialog box appears when you select **Options...** on the **File** menu



>Auto Reduce Fonts

This ensures the fill font will reduce automatically to fit inside the field, up to the number of points specified. Thus, if Auto Reduce Fonts is set to 4, a 10 point font will be reduced successively to 9 point then 8 then 7 then 6 in order to fit text into a space. Once the maximum 4 point reduction is reached, the font will not reduce further.

>Form Backup

Designer can automatically backup forms you create to avoid losing your work. Forms saved with Auto Backup are stored with the “.bak” extension in your eForms directory.

>To set Auto Backup

1. On the **File** menu, click **Options**.
2. In the **Options** dialog box, click the **Create Backup Copy** checkbox.
3. Click the **OK** button.

Note: Auto-Backup is not a timed function.

Each time you save your work, the old instance is saved with a “.bak” extension.

Setting Form Zoom

Visual eForms Designer provides options for resizing the view of the form that you are creating.

>To zoom, do either of the following

On the **View** menu, click **Page, Width, 200%, 100%, or 50%**.

-or-

On the **Standard** toolbar, select one of the following from the drop-down menu:

- **Page, Width, 200%, 100%, 50%, or 25%.**



Adding and Removing Pages

You can add a page to your document before or after the current page.

>To add a page

On the **Page** menu, click either of the following:

- **Add Before Current** to add a page *before* the current page.
- **Add After Current** to add a page *after* the current page.

>To delete the current page

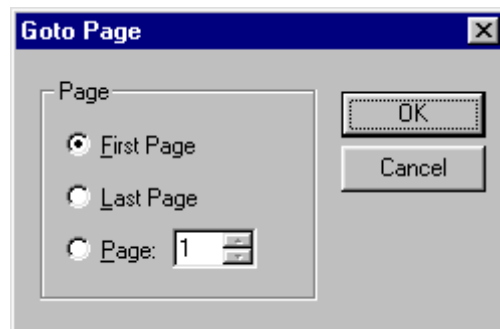
On the **Page** menu, click **Delete**. You will be asked to confirm the deletion.

GoTo Page

You can create forms that have up to 99 pages. In multiple page forms you can add or delete pages. The **Goto** page function allows you to jump to a specific page.

>To go to a specific page in a form

1. On the **Page** menu, click **Goto**.



2. When the dialog box appears, you have the following options:

- Click the **First Page** option icon to go to the first page of a document.
- Click the **Last Page** option icon to go the last page of a document.
- Click the **Page** option icon and scroll to find the desired page or enter the desired page number directly.

3. Click the **OK** button.



>To go to the previous page in a document

Click the **Previous Page** icon on the **Standard** toolbar.



>To go to the next page in a document

Click the **Next Page** icon on the **Standard** toolbar.



>To go to the first page in a document

Click the **First Page** icon on the **Standard** toolbar.



>To go to the last page in a document

Click the **Last Page** icon on the **Standard** toolbar.



Undo

The Undo feature allows you to cancel a previous action. There are ten (10) levels of undo. That is, you can undo up to ten consecutive previous actions.

Note: All changes inside a dialogbox such as *Tab Order*, *Accessibility* and *Properties* are regarded as one single action. Therefore, UNDO revert all changes user makes while inside a dialogbox.

Click the Undo button as many times as needed.

Clicking the opposite arrow enables you to Redo the most recent Undo action.

Setting Form Properties

To set the properties of a form, you must open the **Properties** Window. The form properties are viewable from the **Properties** Window only. To open the Properties window, ensure **Properties** is checked in the **View** menu:

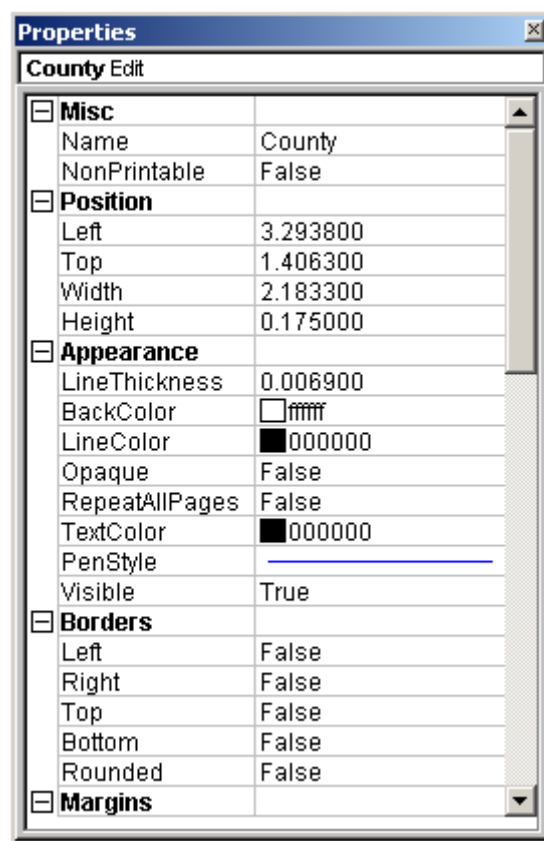
On the **View** menu, click **Properties**. This will hide/show the **Property** control bar.



Property control bar handle

To display the **Properties** Window, do one of the following:

- On the **View** menu, click **Toggle Properties Mode**
- Double-click on the **Property** control bar *handle*
- Click the **Property** control bar *handle* and drag it to the form window.
- Press [ALT] + [M].



In this properties window, you can enter information such as form name, description, author, etc. This information helps categorize the form and can be used in many ways. Examples: Enterprise Server or other tool to develop a forms catalog. As part of an document management system for archived retrieval. Version tracking.

Property	Description
Name	Form name.
Description	Form description.
Version	The form version must be set if the form will be used with Visual eFlow Enterprise Server.
Author	Name of the form creator.
Category	Category for the form (if applicable).
Keywords	Keywords for locating the form in a search engine.
Copyright	Form copyright year.
Comments	Any form comments.
IndexFields	Selects fields that will be indexed.
ArchiveFormat	The file format for archiving the form.
AllowContentSearch	Allows users to search content.
TrackHistory	Allows users to track form history.
Modified	Date the form was modified or created.
Width	Page width; can also be set in Form Setup.
Height	Page height; can also be set in Form Setup.
BackColor	Color of the form's background.

Import/Export

The Import and Export functions allow you to use Designer with other programs. FormFlow 1.x and 2.x files can be imported into designer. Forms created in Designer can be exported as PDF and html files.

>To import a FormFlow file

1. On the **File** menu, point to **Import** and then click **FormFlow**.

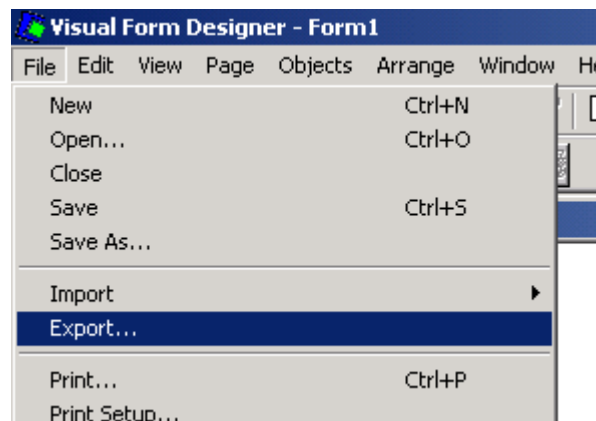
The Open File dialog box appears.

2. Select the appropriate FormFlow file (.frp) and click the **Open** button.

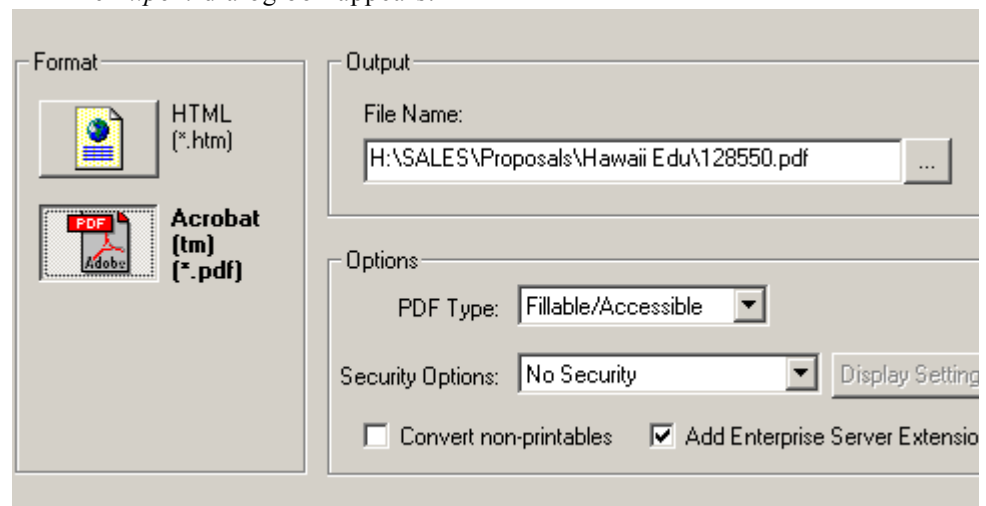
The FormFlow file is imported into your Designer document.

>To export a form

1. On the **File** menu, point to **Export**.



The Export dialog box appears.

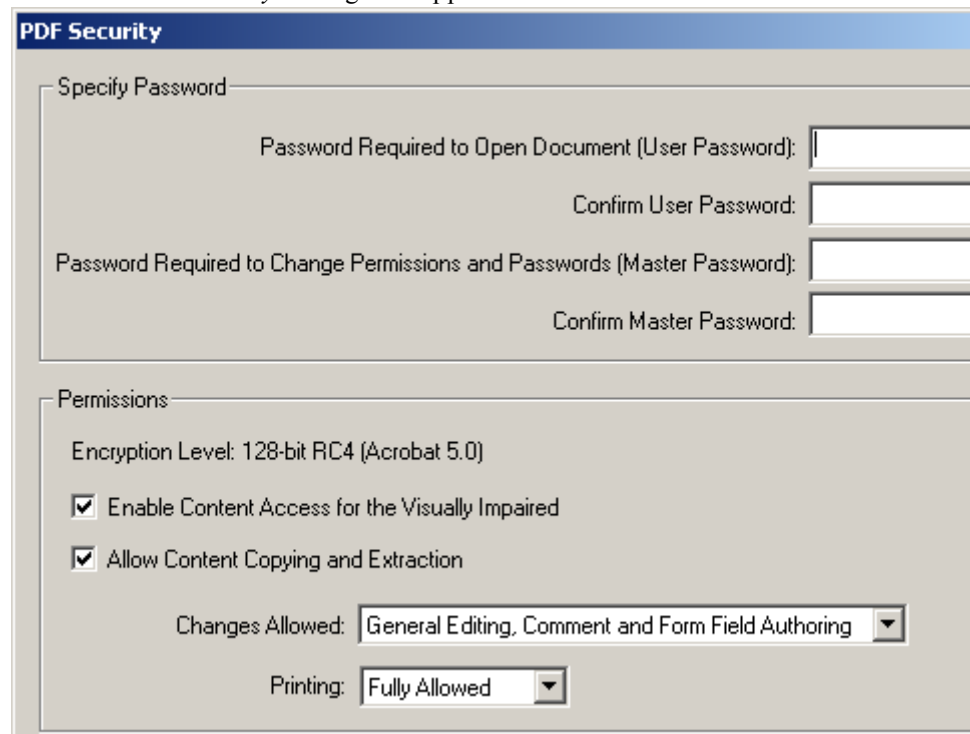


When exporting an accessible form, be sure to select accessible PDF.

2. Select the output format: HTML or PDF.
3. Name the file and select a location in which to save the form.

>Exporting to PDF

1. Select the **Acrobat** button
2. Select the **PDF Type** (one of Flat, Form+Data, Fillable and Fillable/Accessible)
3. Select the **Security Option** (Default is **No Security**)
the *PDF Security* dialog box appears.



The screenshot shows the 'PDF Security' dialog box. It has a title bar 'PDF Security' and two main sections. The first section, 'Specify Password', contains four password fields: 'Password Required to Open Document (User Password):', 'Confirm User Password:', 'Password Required to Change Permissions and Passwords (Master Password):', and 'Confirm Master Password:'. The second section, 'Permissions', shows 'Encryption Level: 128-bit RC4 (Acrobat 5.0)'. It has two checked checkboxes: 'Enable Content Access for the Visually Impaired' and 'Allow Content Copying and Extraction'. Below these are two dropdown menus: 'Changes Allowed:' set to 'General Editing, Comment and Form Field Authoring' and 'Printing:' set to 'Fully Allowed'.

>Exporting to HTML






1. Select the **HTML** button
2. Click the **Save** button.
The form is saved as an html document and can be opened with any web browser program.

Working With Objects

The electronic forms that you create are comprised of the objects that you draw. They can contain text, graphics, or other design elements of your choice. The **Object** toolbar allows you to draw both fillable and non-fillable objects.

Non-Fillable Objects

Non-fillable objects are designed *not* to be modified by a user in the **Filler**. The following tools are used to draw non-fillable objects:















Icon	Name	Use To...
	Text	create a text object where fill text cannot be added.
	Box	draw squares, rectangles, and round cornered box objects.
	Circle	draw circles and ellipses.
	Image	incorporate an embedded or linked image into a form.
	Line	draw straight lines in any direction.



To draw a straight line, perfect square or perfect circle, hold [ALT] while drawing the object

Fillable Objects

Fillable objects are designed for users to enter information in the **Filler**. The following tools are used to draw fillable objects:

Icon	Name	Use To...
	Hyperlink	add a hyperlink field to the form.
	Edit Field	create a fillable text and number object.
	Number	create a fillable number object.
	Currency	create a fillable currency object.
	Date	create a fillable date object.
	Mask	create a fillable mask object, format defined by designer.
	Check Box	create a fillable check box object.
	Button	create a button to execute a macro.
	Editable Image	create an image where the user types the pathname for the image in the Filler .
	Drop List	create a scrolling drop list.
	Table	create a fillable table object.
	Bar Code	create bar code object.
	Digital Signature	add a digital signature to the form.
	Rich Text	add a rich text object.

Drawing Objects

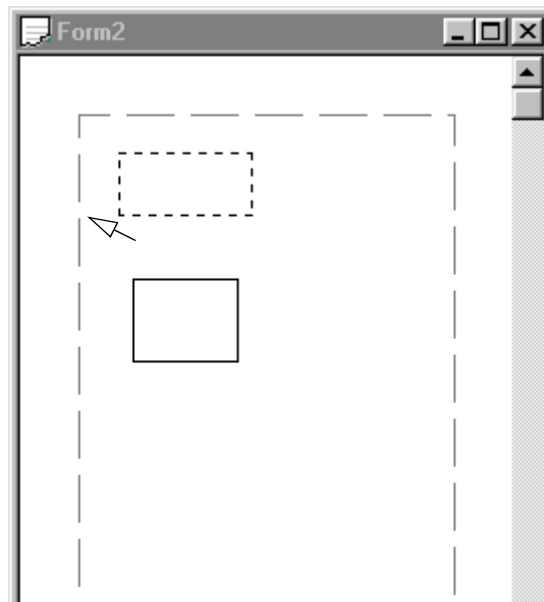
Identical steps are used to create most objects; they are drawn the same way whether they are fillable or non-fillable.

>To draw an object

1. Click the desired object on the **Object** toolbar.



2. Move the cursor into the form window where you would like to place the object.
3. Hold down the mouse button and drag to draw the desired object in the form window.



4. When the object is the size you want, release the mouse.

The object is displayed in the form window.

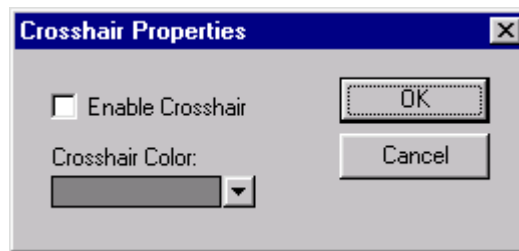
Crosshairs

When drawing or positioning an object, you can use crosshairs to precisely measure the size of the object.

>To enable crosshairs

1. On the **View** menu, click **Crosshair**.

The Crosshair Properties dialog box appears.



2. Check the Enable Crosshair box and select a crosshair color from the drop-down menu (or it can be left as the default grey color).
3. Click the **OK** button when finished.

The pointer now includes a vertical and horizontal crosshair.

Positioning Objects

Creating an effective form requires the ability to position objects precisely. Visual eForms Designer provides you with a variety of tools and guides to help you do this. These include a ruler, grid, and position and align commands.

Selecting Objects

To position objects, you must first select those objects.

>To select an object

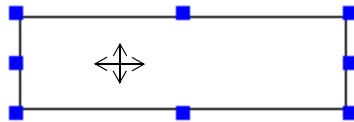
Click on the object.

-or-

Hold down the mouse button and drag the pointer around or through the object, thus creating a *rubberband*. This selects any object touched by the rubberband.

When you draw or select an object, blue boxes surround the outside borders of the object. These are referred to as *handles*. The *handles* reveal the position of the object.

When an object is selected, the pointer is displayed as *crosshairs* inside the selected object.



>To select multiple objects, do any of the following:

- Hold down the [SHIFT] key and click on the objects.
- Hold down the mouse button and drag the pointer around or through the objects to rubberband them.
- On the **Edit** menu, click **Select All**.



By holding [CTRL] while creating a rubberband, only the objects that fall within the rubberband are selected.

>To select special objects

1. On the **Edit** menu, click **Select Special**.
2. In the **Select Special** dialog box, check all of the objects that you would like selected.
3. Click the **OK** button.

>To select tables

1. Click on the table once.
2. Click again inside the table to select an individual table cell.

>To select table columns or table rows


1. Click the table once to select it.
2. Position the cursor on the table border of the column or row you want.
3. The mouse changes to an arrow. Click and all cells in the the column or row are selected.

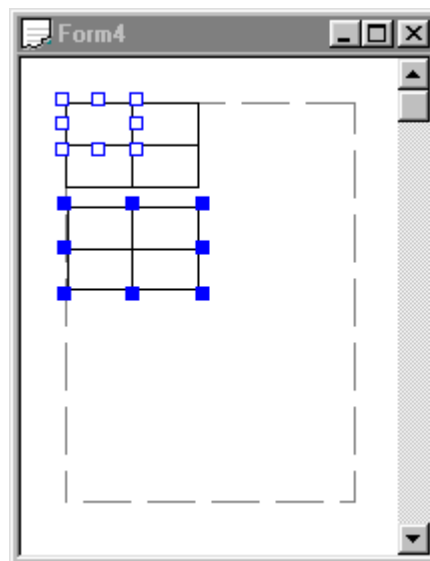
>To change the width of columns or rows

1. Click the table once to select it.
2. Position the cursor on the line between two columns or rows.
3. The mouse changes to parallel lines and an arrow. Click, hold and drag the cursor to change the width of the column or height of the row.

>To select all cells of a table

1. Click the table once to select it.
2. Hold the SHIFT key and click on a blank area outside the form.

 Select multiple tables as you would select multiple objects.

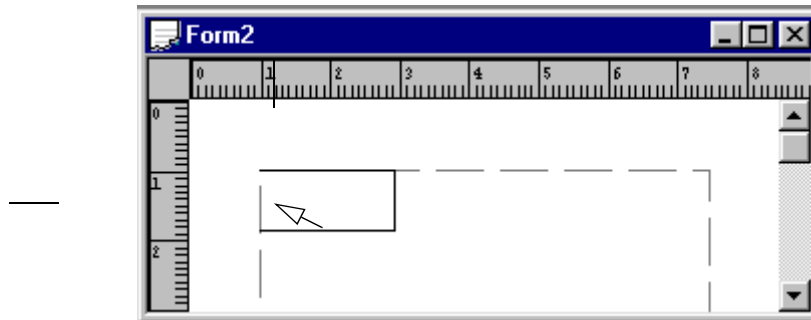
**Ruler**

When the ruler is displayed, the pointer is referenced by guides on the ruler that follow the coordinates of the mouse. This allows you to select an object and drag the handles to position the object at exact coordinates in your form window.

The increments on the rule change according to the Measurement System selected from Form Setup.

>To display the ruler

On the **View** menu, click **Rulers**.

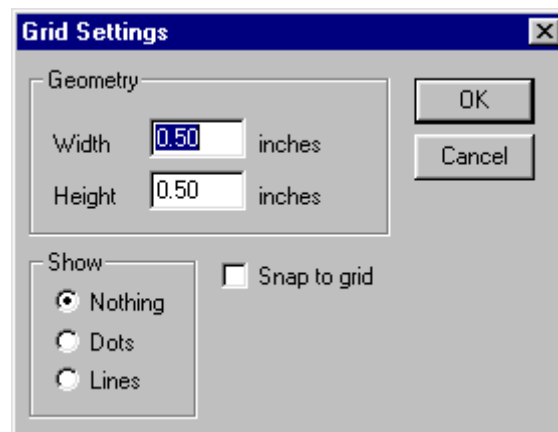


Grid

A grid is a series of intersecting horizontal and vertical lines. When displayed, the grid looks like a sheet of graph paper. You can create a grid with small divisions (geometric settings) for precise placement or large divisions (geometric settings) for more general placement. The grid allows you to position objects in your form window.

>To display the grid

1. On the **Arrange** menu, click **Grid**.



2. In the **Geometry** section, enter a value for the **Height** and **Width** of the grid coordinates.

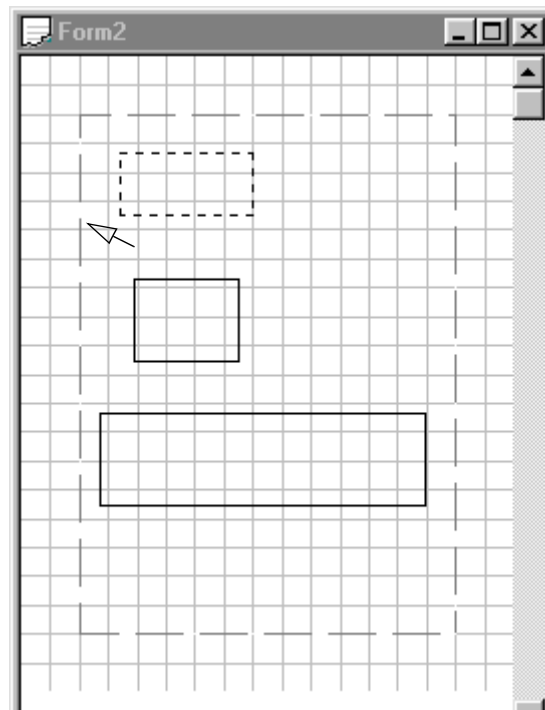
The smaller the value, the smaller the grid boxes.

3. In the **Show** section, click a radio button to show **Lines**, **Dots**, or **Nothing**.

Lines display intersecting lines; **Dots** display dots at every intersection.

4. If you wish, enable **Snap to grid**. See the next section for a full explanation of this feature.

5. Click the **OK** button.



The grid with lines is displayed in the form window.

Using Snap to Grid

A grid is a set of horizontal and vertical lines that act as a guide when moving or resizing objects. Each dot or line intersection represents a point on the grid.

When you enable **Snap to grid**, objects are automatically shifted to the nearest point on the grid as they are resized, created or moved.

>To test the snap to grid feature

1. Create a grid using the previous instructions.
2. In the **Grid Settings** dialog box, make sure that **Snap to grid** is selected.
3. Draw an object.
4. Select the object and drag the handles to resize the object.

The object borders *snap* to align with the grid lines.

Note: *The object's size is influenced by the size of the grid. The borders snap to align with the grid lines.*

>To test objects created without the grid

1. In the Grid Setting dialog box, make sure that Snap to grid is not selected.
2. Draw an object.
3. Turn the grid on.
4. Move the object. Notice that the original object size remains but the borders snap to align with the nearest grid.
5. Resize the object. Notice that the object snaps to align with the nearest grid.

Aligning Objects

You can align selected objects by their left or right sides, top or bottom edges, or align the object centers either vertically or horizontally. All objects are aligned with the last object selected. The last object selected has solid blue handles; the other objects have hollow blue handles.

>To align objects

1. Select the objects.
2. On the **Arrange** menu, click **Align**.
3. Select one of the following options:
 - **Left**, to align the objects on the left side.
 - **Horizontal Center**, to align the objects by their horizontal center.
 - **Right**, to align the objects by their right side.
 - **Top**, to align the objects by their top side.
 - **Vertical Center**, to align the objects by their vertical center.
 - **Bottom**, to align the objects by their bottom side.



For more information about the Standard toolbar, see "Standard Toolbar" on page 9.

Sizing Objects to Match

You can quickly make two objects exactly the same size. All objects are sized with the last object selected.

>To make objects the same height or width

1. Select the desired objects.
2. On the **Arrange** menu, do one of the following:
 - point to **Make Same Size** and then click **Width**
 - point to **Make Same Size** and then click **Height**

-or-

On the **Standard** toolbar,



- click the **Same Width** icon.
- click the **Same Height** icon.

Layering overlapping objects

You can move overlapping objects from front to back or back to front.

>To layer overlapping objects

1. Select the object.
2. On the **Objects** menu, click **Bring to Front** or **Send to Back**.

Modifying Objects

When an object is selected, you can modify it. When resizing and moving, the cursor changes shape, depending on where the cursor is placed.

>To resize an object

1. Select the object.
2. Click any handle on the selected object.
3. Hold down the mouse button and drag to resize the object as needed.
4. Release the mouse button.

>To move an object

1. Select the object.
2. Hold down the mouse button and drag the selected object to another location.
3. If desired, constrain the movement to horizontal or vertical by holding [SHIFT] while dragging the object.
4. Release the mouse button.

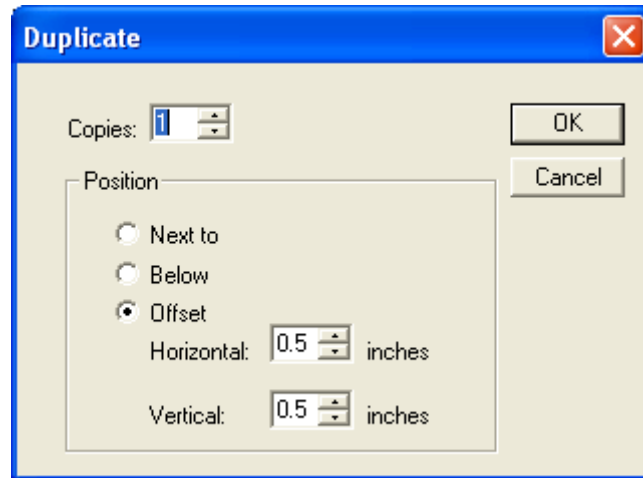
>To copy an object

1. Select the object.
2. On the **Edit** menu, select **Copy**.
- or -
press **CNTL+C** on your keyboard.
3. On the **Edit** menu, select **Paste**.
- or -
press **CNTL+V** on your keyboard.
4. Hold down the mouse button and drag the copy to a new position.

>To duplicate an object

1. Select the object.
2. On the **Edit** menu, select **Duplicate**.
- or -
press **CNTL+D** on your keyboard.
3. Enter the number of duplicate copies.
4. Determine the position
 - Next to, which positions the copy immediately to the right of the original with adjacent edges touching
 - Below, which positions the copy immediately below the original with adjacent edges touching

- Offset horizontal, which positions the copy on the right side of the original a specified distance from the left edge of the original object. If more than one duplicate is specified, subsequent copies are placed the same distance from the left edge of the previous copy.
- Offset vertical, which positions the copy a specified distance from the top edge of the original object. If more than one duplicate is specified, subsequent copies are placed the same distance from the top edge of the previous copy.



>To delete an object

1. Select the object.
 2. On the **Edit** menu, select **Delete**.
- or -
- press the **Delete** key on your keyboard.

>To group objects

1. Select the objects.
2. On the **Objects** menu, select **Make Group**.

>To break a group of objects

1. Select the group of objects.
2. On the **Objects** menu, select **Break Group**.

>To convert an object into another type of object

1. Select the object.
2. On the **Edit** menu, click **Convert**.
3. Click the radio button that corresponds to the desired object.
4. Click the **OK** button.



When changing properties of grouped objects, only common properties are available for change.

Creating an Object Library

An object library is a collection of form objects that you have created and want to use repeatedly. This can be an individual field or a group of fields. The desired objects are placed into the Object Library by drag and drop. You can give each object a name. You can scroll through the objects in the library and then copy them to your form.

>To add form objects to the Object Library

1. Create the object.
2. On the **Objects** menu, click **Object Library**.
3. Click on the object you want to add to the **Object Library** and drag it to the **Object Library** window.

A copy of the object will be placed. The original on the form is not affected.

4. Right click on the object in the **Object Library** and select **Rename**.

>To place form objects from the Object Library

1. On the **Objects** menu, click **Object Library**.
2. Scroll through the list of objects and click on the object you want to add to the form.
3. Drag the object to the form.

*A copy of the object remains in the **Object Library**.*

>To delete form objects from the Object Library

1. On the **Objects** menu, click **Object Library**.
2. Scroll through the list of objects and click on the object you want to delete.
3. Right click and select **Delete**.

You can have multiple Object Libraries accommodating varying standard design requirements.

>To create multiple Object Libraries

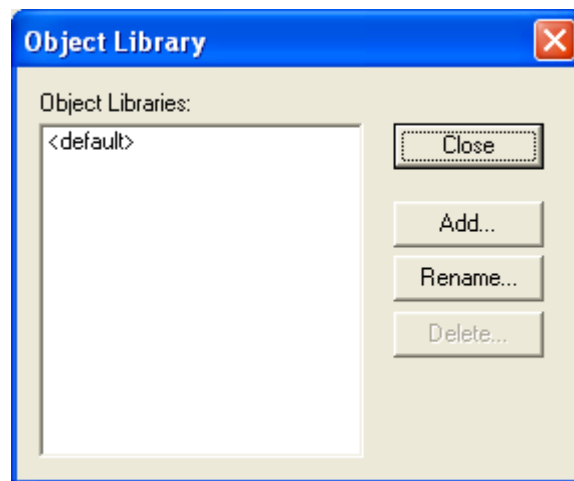
1. On the **Objects** menu, click **Object Library**. Click the ellipsis (...) and open



the Object Library dialog box.

- 2.

3. Click the Add button and enter the name of the Object Library.



4. Click OK.
5. Return to the Object Defaults dialog box and select Set.
The name of the active object default group now appears in "Active Defaults."
6. Click Close.
7. *New default properties are ready to be created within the new defaults group. Go to "set default object properties" (above) to change the default properties on the individual objects.*

Changing Object Properties

All objects have common properties that can be changed. Changing their properties changes the way they look and function in the form. Not every object has every property. Some, like masks, have additional properties. This section explains how to change properties in objects.

Using the Properties Windows

To change the properties of an object, you must first open the **Properties** window, which contains property sections such as Appearance, Borders, Calculations, Color, Fill Characteristics, Font, Justify, Lines, Margins, Name, Position, and Text.

>To open the Properties window

To set the properties of a form, you must open the **Properties** Window. The form properties are viewable from the **Properties** Window only. To open the Properties window, ensure **Properties** is checked in the **View** menu:

On the **View** menu, click **Properties**. This will hide/show the **Property** control bar.



Property control bar handle

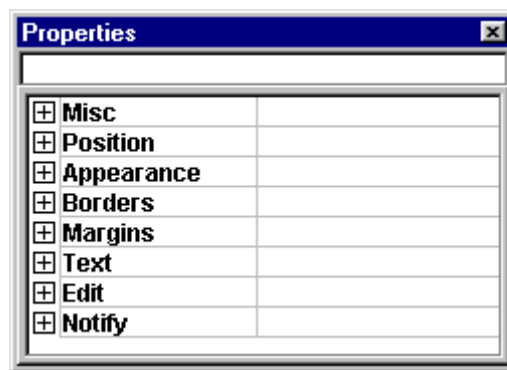
To display the **Properties** Window, do one of the following:

- On the **View** menu, click **Toggle Properties Mode**
- Double-click on the **Property** control bar *handle*
- Click the **Property** control bar *handle* and drag it to the form window.
- Press [ALT] + [M].

The **Properties** window appears.

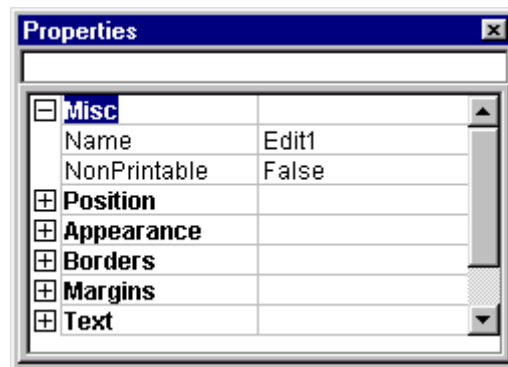


To display the alternative Properties window, right-click an object. See "Alternative Properties Window" on page 80 for more information.



>To change object properties

1. Select the object to be changed.
2. Open the **Properties** window.
3. Identify the **Property** that you would like to change: **Miscellaneous**, **Position**, **Appearance**, **Borders**, **Margins**, **Text**, **Edit**, or **Notify**.

Miscellaneous**>To name the object**

Enter a name for the object in the **Name** field.

The **Name** defaults to whichever object you have selected, such as **Edit**, **Text**, or **Line**, but you may want to change it to a name that is meaningful to you, such as **Label**, **Logo**, etc.

>To designate which objects print on your form

Click the drop-down menu in the **Non Printable** field.

If **True** is selected, the object does not print as a part of the form. If **False** is selected, the object does print.

Setting Default Object Properties

Default object properties can be set for each object; these properties then appear each time you create a new object. The object properties can be set and retained as a group, accommodating varying standard design requirements.

>To set default object properties

1. Click the object icon for which you want to set the default properties.
*The Properties Window is now in **Defaults** mode, as indicated.*
2. Set the properties of your choice in the **Properties** window.

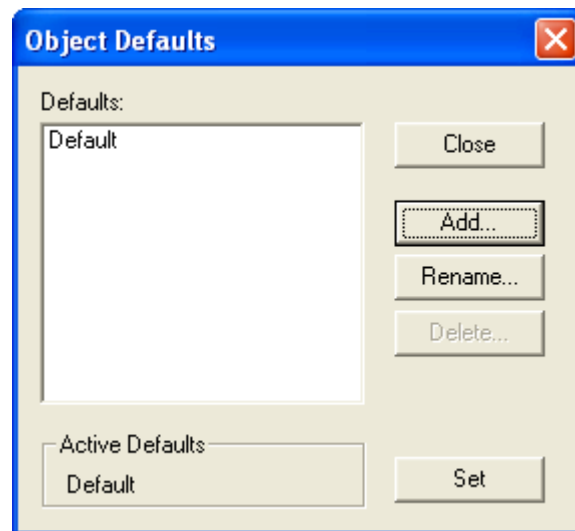
3. When finished, create an object to ensure your default properties have been set.

The default properties are now set on this object. Return to step 1 to change the default properties on another object.

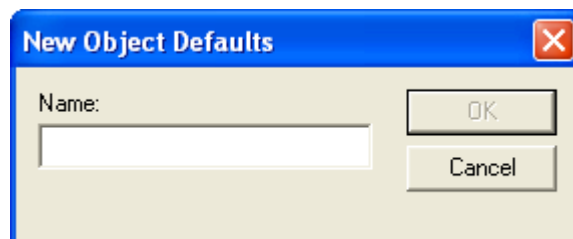
After you set the default properties, every time you create a new object of the same type, it will retain your default properties.

>To create multiple object default groups

1. On the **Objects** menu, click **Object Defaults**.



2. Click the Add button and enter the name of the new object defaults group.



3. Click OK.
4. Return to the Object Defaults dialog box and select Set.

The name of the active object default group now appears in “Active Defaults.”

5. Click Close.

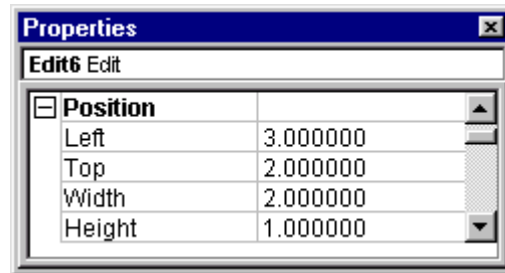
New default properties are ready to be created within the new defaults group. Go to “set default object properties” (above) to change the default properties on the individual objects.

Position

The position of the object details the coordinates of the object and the size of the object.

Left and Top determine where your object is positioned on the page. Width and Height determine the size of the object.

By looking at the Properties window below, you can tell that the selected object is positioned 3 inches from the Left of the page and 2 inches from the Top of the page. It has a Width of 2 inches and a Height of 1 inch.



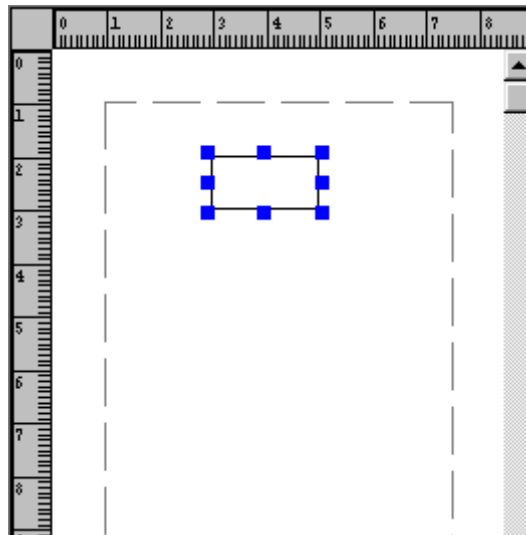
>To position an object by coordinates

Enter values in the **Left** and **Top** fields.

These numbers determine where your object is positioned on the form. Entering "3" in the field Left and "2" in the field Top makes the top left corner 3 inches from the left and 2 inches from the top.



To optimize this feature, make sure that the View Ruler option is selected.



>To change the object to an exact size

Enter values in the Width and Height fields.

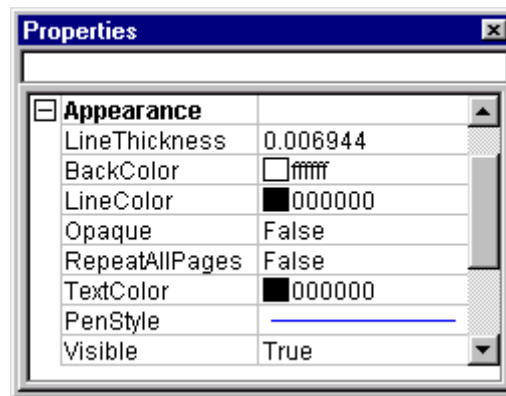
These values determine the size of your object. Entering “2” for the Width and Height values create an object 2 inches high and 2 inches wide.



The best way to align two objects or to make them the same size is to use the numerical values in Position or use the alignment and sizing icons. Eyeballing doesn't work.

Appearance

Appearance refers to the way the object looks on the page, including its colors and outline.



>To change the thickness of the outline of the object

In the LineThickness field, click the drop-down menu and select a line.

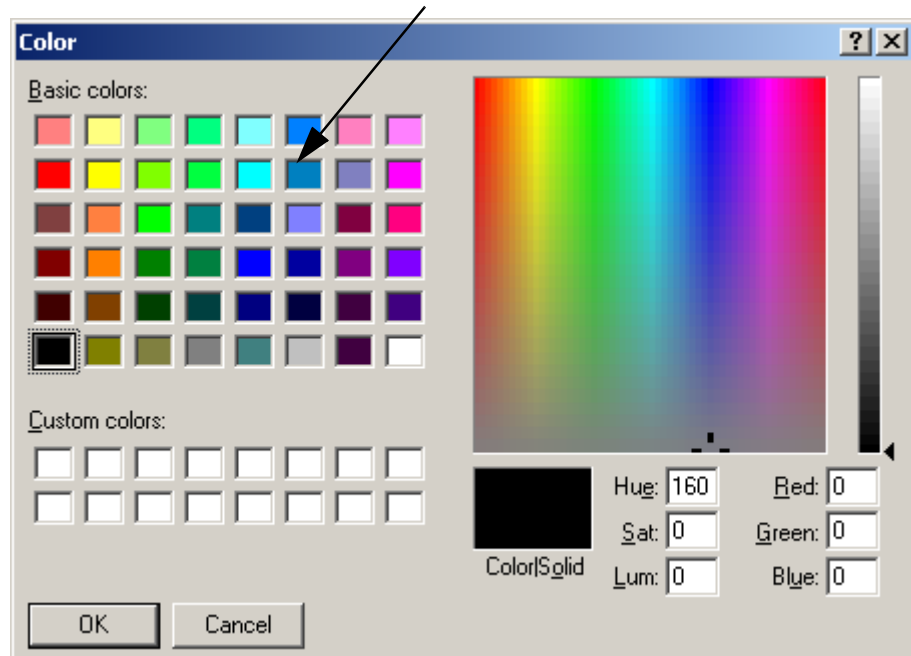
Each line is labeled in fractions of an inch and determines the width of the border around the object.



The selection "1/300" is comparable to a hairline rule. The selection "1/72" is comparable to a 1 point rule.

>To change the object's background color

1. Click the drop-down menu in the **Back Color** field.
2. Select a color from the **Color** dialog box and click the **OK** button.

**>To select a Custom color**

1. Click your cursor on a part of the spectrum window.
2. Click the arrow in the vertical value selector to indicate the amount of white or black that is mixed with your color.
3. The color you have selected will display in the ColorSolid window.
4. Adjust the color and value by moving the cursor or the arrow.
5. Click OK to select this color for your object.

>To change line color

Click on the drop-down menu in the LineColor field to select a color.

>To create opaque objects

Double click Opaque or click the drop-down menu in the Opaque field to select an option.

True makes the object opaque.

False makes the object transparent.

>To have the object appear on every page

Double click Repeat All Pages or click the drop-down menu in the Repeat All Pages field to select an option.

True places the object on every page of the form.



Double click on the property name that has a True/False choice, such as Visible or Opaque. This acts as a toggle to change the existing state. This toggle action works with many property names.

>To change text color

Click on the drop-down menu in the TextColor field to select a color.

>To change the outline style

Click the drop-down menu in the Pen Style field to select a style.

You may choose from a series of solid or dotted lines. **Pen Style** determines if the outline of your object is a solid or dotted line.

>To determine if the object is visible on the form

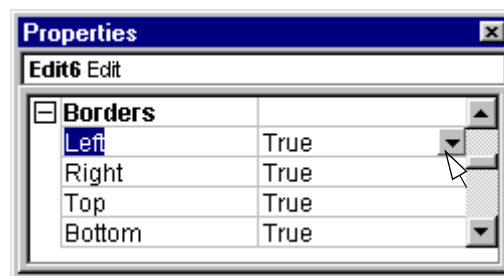
Double click Visible or click the drop-down menu in the Visible field to select an option.

True makes the object visible on the page

False makes the object invisible on the page.

Borders

Borders refer to the outline of the object and how that outline functions.



>To create visible borders

1. Double-click on **Left** or click on the drop-down menu in the **Left** field to select **True** or **False**.

True makes the left border of the object visible, **False** does not.

2. Repeat this step for **Right**, **Top**, and **Bottom** borders, selecting **True** or **False** from the drop-down menu.

3. Double-click on **All Borders** or click on the drop-down menu in the **All Borders** field to select **True** or **False**.

True makes all borders of the object visible, **False** does not.

>To choose square or round corners for the object

Double-click on **Rounded** or click on the drop-down menu in the **Rounded** field to choose **True** or **False**.

True creates rounded borders for the object.

False creates squared corners for the object.

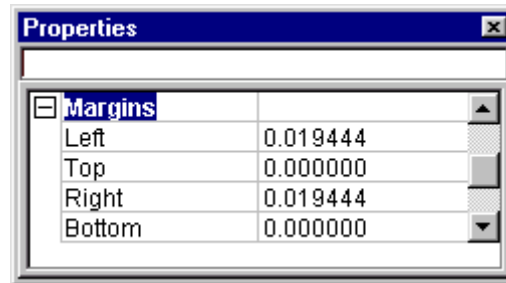


If round corners are selected, all borders are automatically set to **True**. If the background color of a form is white and the object borders are white, a border can be set to **True** but still not be visible to the eye.



Margins

Margins determine how text in an object is positioned relative to the borders of the object. The smaller the number, the smaller the space between the text and the border.



>To set the width of the margins

1. Enter a value in the Left field.
2. Repeat in the Top, Right, and Bottom fields.

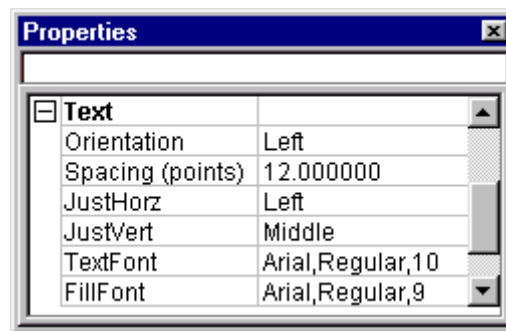


Keep in mind that there are two kinds of Fonts: Text

Font and Fill Font. **Text Font** is used in the text or label portion of a field. **Fill Font** is used for data users enter into a field while in the Filler.

Text

The text property allows you to define the alignment and font of the text in the object.



>To select text orientation

Click the drop-down menu in the Orientation field to select one of the following:

- Left
- Up
- Down

LEFT

UP

DOWN

>To select text spacing

Within a text block, text spacing is measured from the baseline of one line of text to the baseline of the text below it. The default text spacing is 2 points.

Click in the **Spacing** field and enter a value.

This determines the spacing between lines of text or data.

>To select text justification

1. Click the drop-down menu in the **JustHorz** field to select one of the following:

- Left
- Right
- Full
- Center

LEFT LEFT LEFT LEFT LEFT

RIGHT RIGHT RIGHT RIGHT

FULL FULL FULL FULL FULL

CENTER CENTER CENTER

2. Click the drop-down menu in the **JustVert** field to select one of the following:

- Top
- Middle
- Bottom

TOP

MIDDLE

BOTTOM

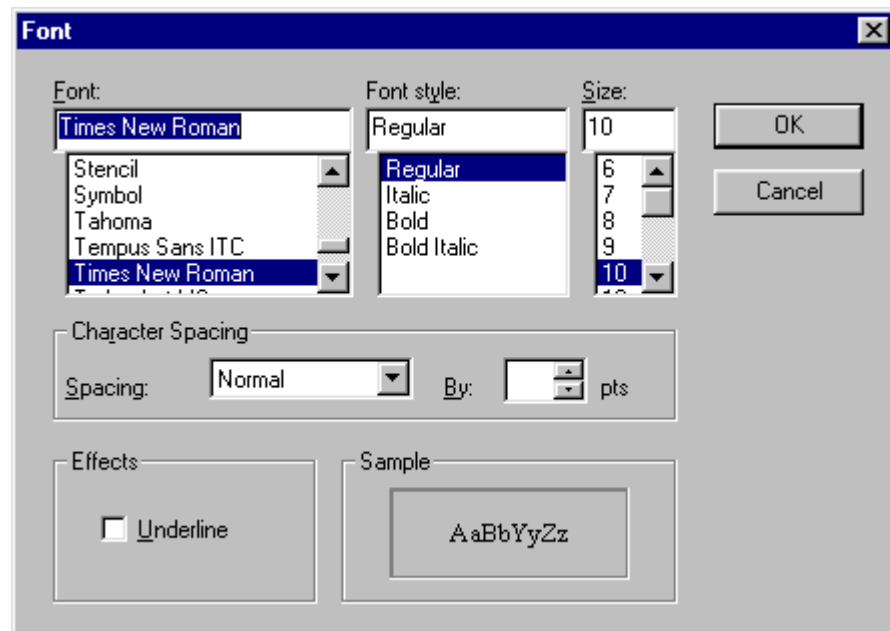
>To select text font

You may choose to have the Fill font different from the

Text font.

1. Click the drop-down menu in the TextFont field.

The Font dialog box appears.



2. In the **Font** section, select the desired font.
A sample is displayed for you.
3. In the **Font style** section, select a font style.
4. In the **Size** section, select a font size.
5. In the **Character Spacing** section, select **Normal**, **Expanded**, **Condensed**, or **Fixed**.
6. If you wish to have the text condensed, expanded, or fixed, click the drop-down menu and select the appropriate number of points.
7. In the **Effect** section, click the **Underline** checkbox if you want underlined text.
8. Click the **OK** button when finished.

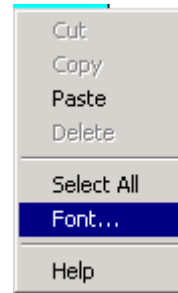
>To select fill font

1. Click the drop-down menu in the FillFont field.

The Font dialog box appears.

2. Complete FillFont values as you completed TextFont values.

Note: While in the Filler, user can change the Fill Font. Simply right-click on a field, and select Font from the drop list.

**Mixed Fonts**

After you have created an object and added text, you can then mix fonts to add style and emphasis. For example, if you'd like to emphasize a point you could display text as shown here:

Sign your name, ***last*** name first

In this example, "last" is a larger font, italicized, and bold.



Font icon

> To mix fonts

1. Select the text you wish to modify and then click the **Font** icon on the **Standard** toolbar.
2. In the **Font** dialog box, make any modifications necessary.
3. When finished, click the **OK** button.

You may repeat this process as often as you'd like, with any text.

Note: You can mix fonts in this manner. You cannot mix text colors in this manner. All text colors within one field must be the same.

RTL

RTL stands for Right to Left. Some languages, such as Arabic, are read right to left.

>To select Right to Left reading

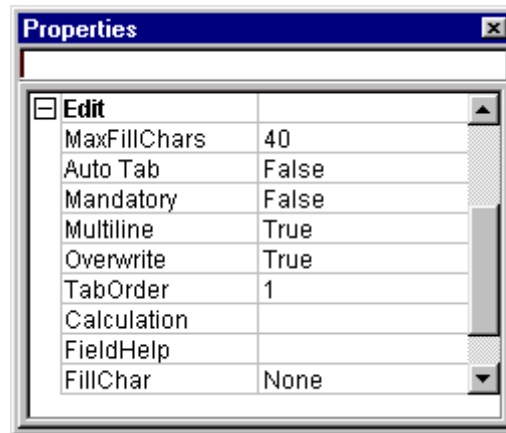
Double click RTL or click the drop-down menu in the RTL to select an option.

True makes the object read from right to left.

False makes the object read from left to right.

Edit

Edit allows you to determine how the fill object functions when the user enters information in the **Filler**.



>To determine the number of text characters allowed in an object

Enter a value in the MaxFillChars field.

MaxFillChars determines the maximum number of characters that a user can type while in the **Filler**.

>To change auto tab

Double-click **Auto Tab** or click the drop-down menu to select **True** or **False**.

When the user is in the **Filler**, **Auto Tab** causes the cursor to jump automatically to the next field as soon as the user enters valid information.

True enables **Auto Tab**, **False** disables it.

>To change mandatory

Double-click **Mandatory** or click the drop-down menu to select **True** or **False**.

When the user is in the **Filler**, **Mandatory** prevents the user from moving to the next field until valid information has been entered. Note: This property is not active while the designer is testing, so the results of **True** are not immediately obvious.

True enables **Mandatory**, **False** disables it.

>To change multiline

Double-click **Multiline** or click the drop-down menu to select **True** or **False**.

When the user is in the **Filler**, **Multiline** allows the user to press [ENTER] in a field in the **Filler**. **True** enables **Multiline**, **False** disables it.



Overwrite allows a user to overwrite the default values that you have assigned.

>To enable overwrite

Double-click **Overwrite** or click on the drop-down menu to select **True** or **False**.

Overwrite allows a user in the **Filler** to enter a field and change its value.

True enables **Overwrite**,

False disables **Overwrite**.

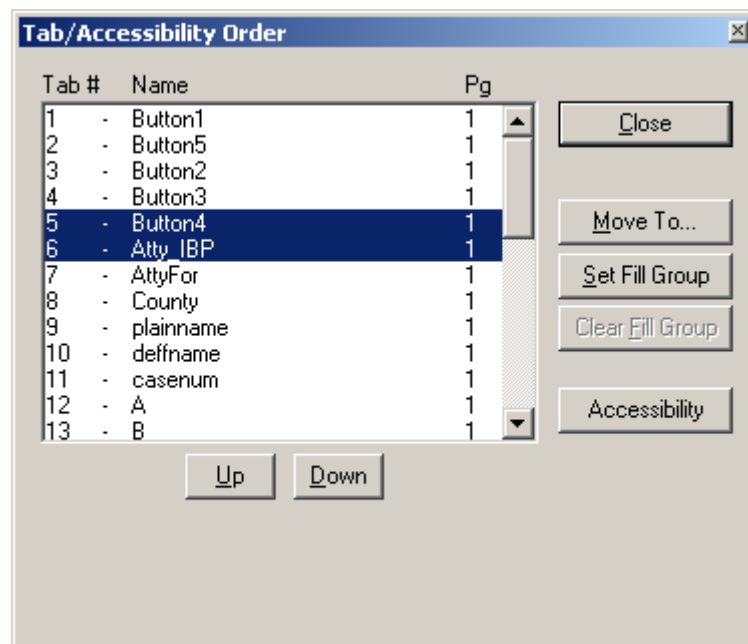
Note: Setting **Overwrite** to **False** is useful when you have a fillable object with a pre-determined value and it is used in a calculation.

>To change the tab order

By default, the first object *drawn* on the form is at Tab order 1. To change this, you must change the tab order.

1. Click the drop-down menu in the **TabOrder** field.

The **Tab Order** dialog box appears.



2. Select the object to be changed and do either of the following:
 - Click either **Up** or **Down**, depending on the position of the object.
 - Click **Move To** and enter the position in the **New Tab Order** field.
3. Click the **Close** button when finished.

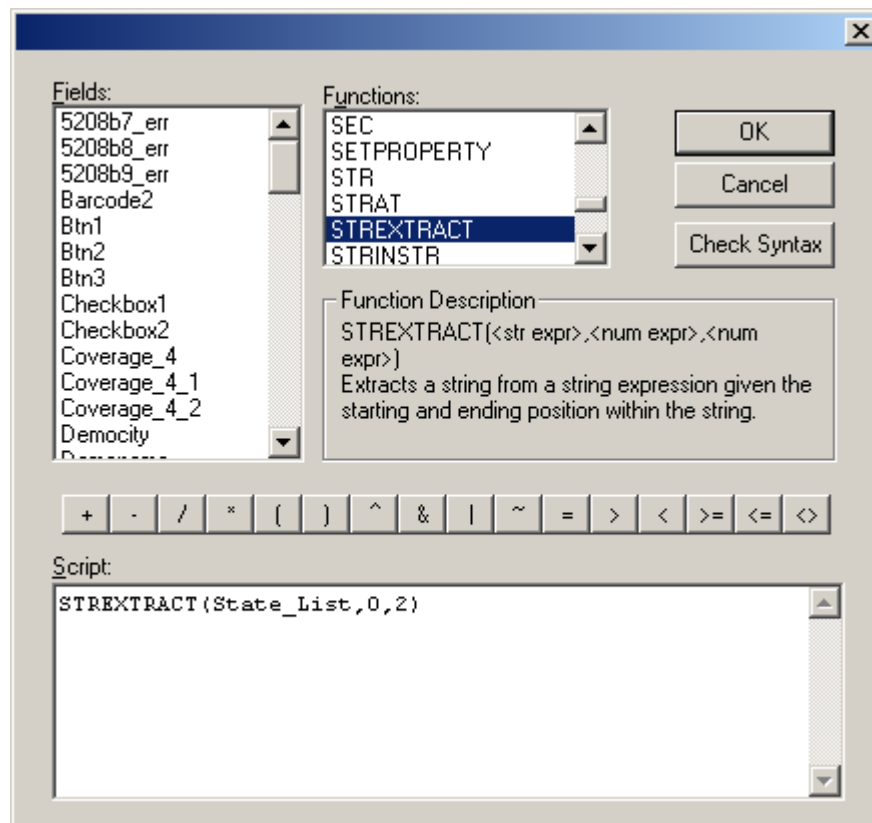
>To enter a calculation

1. Click the drop-down menu in the **Calculation** field.

The Calculation dialog box appears.

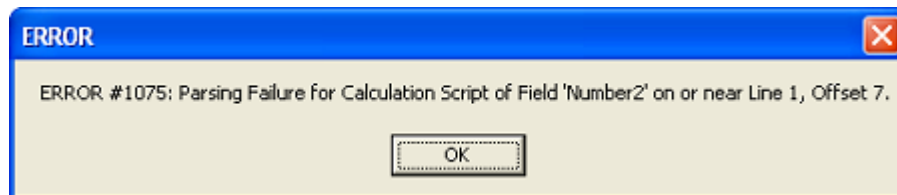


For more information on creating calculations, see "Scripts" on page 94



2. Double-click the **Field** in which to place the calculation.
3. Double-click the **Function** that you would like in the **Field**.
4. Click any **Operators** necessary for the **Function**.
5. Click "Check Syntax" to determine if your syntax is correct. If not, a message indicating the problem will appear.

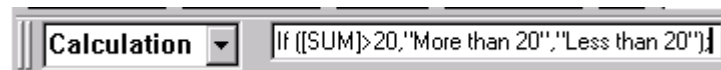
In the error message, *Offset* refers to the number of characters from the beginning of the line where the error has occurred.



The **Fields**, **Functions**, and **Operators** appear in the **Calculations** field.



If you are familiar with calculations syntax, you can quickly enter the calculation in the Control Bar as shown to the right.



>To create field help

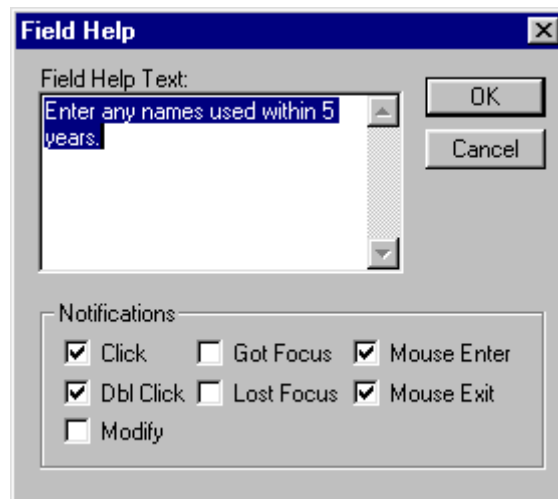
Field help allows you to create a message that appears at fill time.

In order to add Field Help to a field follow these instructions:

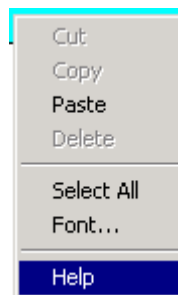
1. Click the drop-down menu in the **FieldHelp** field.
2. When the dialog box appears, enter your message in the **Field Help Text** field.



To view Field Help in the Filler, right click on the field, select the Help option, and the Field Help dialog appears. The Filler application generally decides the manner in which the Field Help displays.



3. In the **Notifications** section, select actions to execute the help message, such as click, double-click, etc.
4. Click the **OK** button when finished.





If you enter 2 as the fill character, the field will be filled with

2's.

The best fill character is an underscore.

>To create a fill character

1. Click the drop-down menu in the **FillChar** field.
2. When the dialog box appears, enter a character to appear in the field while in the **Filler**.
3. Click the **OK** button.

>To view the default value

Enter a value in the **Default Value** field. Text entered in this field is displayed as the default value in the **Filler**.

Note: The *Default Value* can be overwritten in the *Filler* if *Edit/Overwrite* is *True*. The "find and replace" feature does not find text in the *Default Value*.

Notify

The **Notify** properties are referred to as **Notify Flags**. If the **Notify Flags** for a fillable field are set to **True**, a **Notify Message** is sent to the **Filler** application that controls the form.

Notify is processed only in the application and not in the **Filler** itself. Because the action occurs only in the application, nothing can be tested in Preview Mode. On the form, the only response is true or false.

For example, if the **Click Notify Flag** for a given field is set to **True**, the **Filler** application receives a **Notify Message** when a user clicks into that field.

>To change the notify flags

1. Double-click **Click** to select **True** or **False** from the drop-down menu.
 - When the user clicks into this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
 - Available in the **Filler** application.
2. Double-click **Modify** to select **True** or **False** from the drop-down menu.
 - When the user modifies this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
 - Available in the **Filler** application.
3. Double-click **DbClick** to select **True** or **False** from the drop-down menu.
 - When the user double-clicks into this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
 - Available in the **Filler** application.
4. Double-click **GotFocus** to select **True** or **False** from the drop-down menu.
 - When the user's mouse hovers into this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
 - Available in the **Filler** application.
5. Double-click **LostFocus** to select **True** or **False** from the drop-down menu.
 - When the user's mouse hovers out of this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
 - Available in the **Filler** application.
6. Double-click **MouseEnter** to select **True** or **False** from the drop-down menu.
 - When the user enters this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
 - Available in the **Filler** application.

7. Double-click **MouseExit** to select **True** or **False** from the drop-down menu.

- When the user exits this field, the **Notify Message** is sent if the **Notify flag** is set to **True**.
- Available in the **Filler** application.

Special Properties

Fillable objects with special properties include Hyperlinks, Masks, Check Boxes, Buttons, Images, Drop Lists, Tables, Bar Codes and Digital Signatures.

Although these fillable objects have special properties, they are drawn the same as any other object.

>To draw a fillable object

1. Click a fillable object icon on the **Object** toolbar.
2. Move the cursor into the form window where you would like to place the object.
3. Hold down the mouse button and drag to draw the object on the form.
4. When the object is the size you want, release the mouse.

The outline of the object is displayed in the form window.

Hyperlink

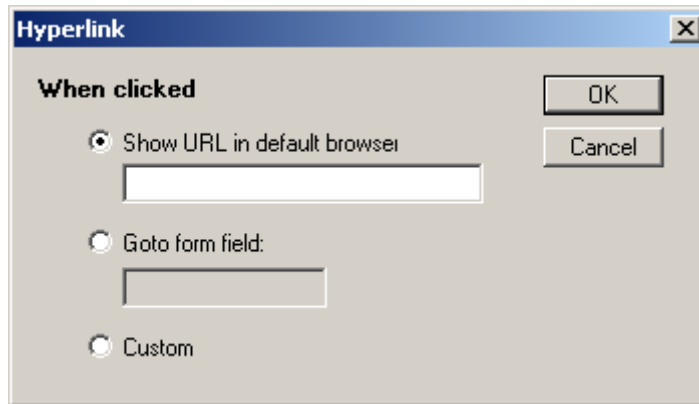
Hyperlinks are used to insert an electronic link that provides access from one document to another or from within one document to another place in that same document.

When you move the mouse over this type of field, the cursor changes to a **Hand** image. If you want the text to become blue or underlined, you must do this at design time.

>To define a URL LinkType

1. Click the drop-down menu in the LinkType field under the **Edit** heading.
2. When the object is clicked, you must define the correct action:
 - To visit a web page select the “*Show URL in default browser*” option and type the URL in the area provided. When the user clicks on the the hyperlink field, the user’s default browser is launched, and the web page referenced by is visited.
 - To move to another field on the same form, select the “*Goto form field*” option and type the name of the field in the area provided. When the user clicks on the hyperlink field, the user’s cursor is placed on the field specified. If the field does not exist, then the hyperlink field will not change focus.

- To handle user-click programmatically, select the *Custom* option. Then a notify event, if one is set, is generated for the **Filler** application to act on.



Masks

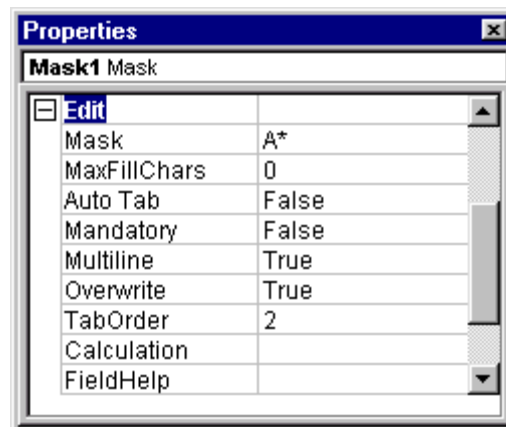
A mask formats data to match a standardized, predefined format. It is useful for fields such as telephone numbers, Social Security numbers, employee numbers, accounting codes that require a standard format. Masks also eliminate typing characters such as - or / when entering this type of data.

When you enter data in a mask, Visual eForms Designer automatically organizes the data in a manner that can include the following:

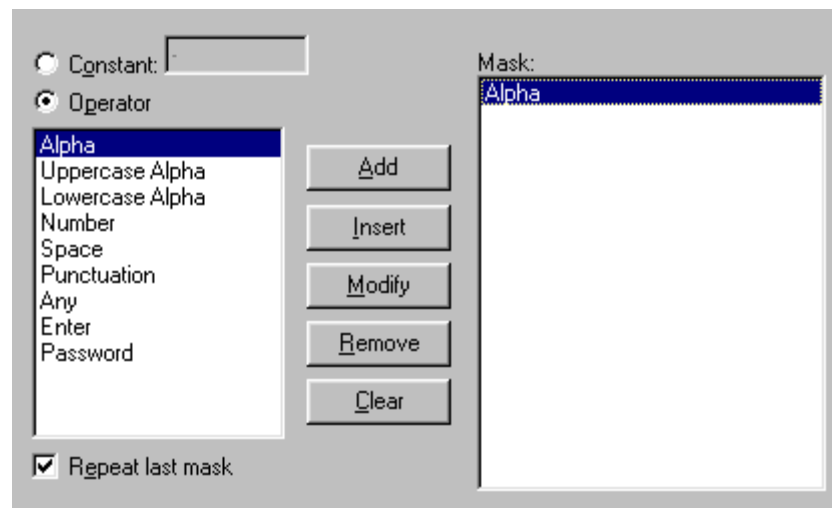
- Inserting constant symbols or characters
- Converting text to upper or lower case
- Adding spaces between characters

>To define the properties in a mask

1. Select the mask object.
2. Open the **Properties** window.
3. Click the drop-down menu in the **Mask** field under the **Edit** heading.



The **Mask** dialog box appears.



Constants that are built into a mask will always appear, regardless of the specific data entered into the field. A constant is any individual character.

4. Click the **Constant** radio button and enter a character in the field.
5. Click the **Add** button to place the constant in the **Mask** field.
6. Click the **Operator** radio button.

Operators are the placeholders in a field mask, used for the variables in the mask that the user will enter.

7. Click the **Add** button to place the operator in the **Mask** field.

The **Constant** and **Operator** appear in the **Mask** field.

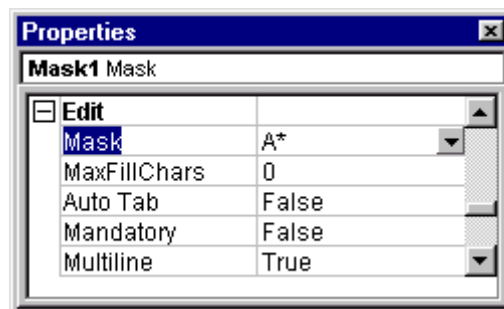
8. Repeat steps 4 - 7 as often as necessary to create the Mask.
9. Click the **OK** button to create the mask.

Example of a Mask

>To create a mask for a Social Security Number

1. Click the **Mask** button on the **Object** toolbar.
2. Draw the Mask object.
3. Open the **Properties** window.
4. Click the drop-down menu in the Mask field.

A = Alpha
U = Uppercase alpha
L = Lowercase alpha
= Number
S = Space
! = Punctuation
. = Any

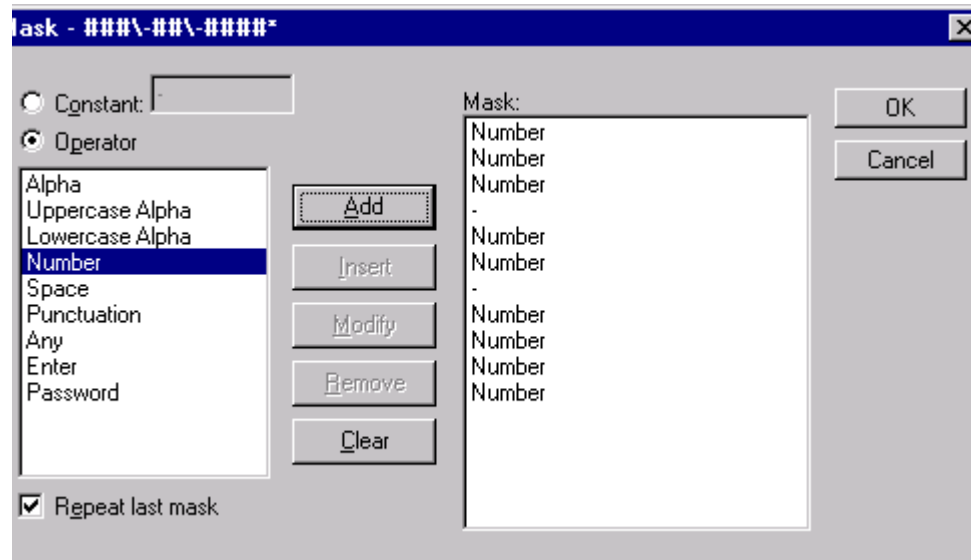


The Mask dialog box appears.

5. Click the **Constant** radio button and enter a **-** in the **Constant** field.
6. Click the **Operator** radio button.
7. Click **Clear** if there is any text in the **Mask** field.
8. Select **Number** and click the **Add** button three times.
9. Click the **Constant** radio button.

As you enter the operators or constants, watch the top of the dialog box. You can see the mask being created there as well.

Once you know the coding, you can add it directly to the Edit/Mask property without going to the dialog box.



10. Click the **Add** button.
11. Click the **Operator** radio button.
12. Select **Number** and click the **Add** button twice.
13. Repeat this sequence until the **Mask** field contains the entire **Number** and **Operator** sequence for a Social Security Number mask object.
14. Click the **OK** button.



If you directly modify the Mask property, insert \ before each constant in the mask sequence.

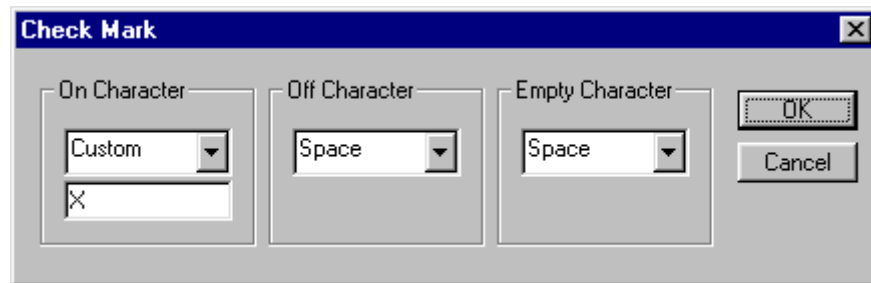
Check Boxes

Use the **Check Box** object to create Fill objects that can toggle between two or more different options such as: Yes/No/Undecided, On/Off, True/False, Correct/Incorrect, Male/Female, etc.

>To define the properties of the checkbox

1. Open the **Properties** window.
2. Click the drop-down menu in the **StrOn** field under the **Edit** heading.

The **Check Mark** dialog box appears.



3. In the **On Character** section, do one of the following:

- Click the drop-down menu to select a character to appear in the check box in the **Filler**.
- Select **Custom** from the drop-down menu and enter a character in the field.

This value determines what the check box looks like when it is **On**.

4. In the **Off Character** section, do one of the following:

- Click the drop-down menu to select a character to appear in the check box in the **Filler**.
- Select **Custom** from the drop-down menu and enter a character in the field.

This value determines what the check box looks like when it is **Off**.

5. In the **Empty Character** section, do one of the following:

- Click the drop-down menu to select a character to appear in the check box in the **Filler**.
- Select **Custom** from the drop-down menu and enter a character in the field.

This value determines what the check box looks like when it is **Empty**.

6. Click the **OK** button.



If you select an option on the drop list for the character, you must select Wingdings as the fill font.

If you define the character as “custom”, you can use any font you want.

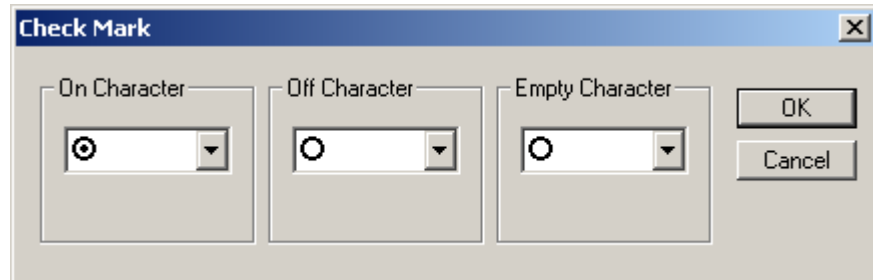


Programmatically, the value of on = 1 and off = 0, or stated another way, true = 1 and false = 0.

This is true for check boxes and other objects that have a true/false option.

Radio Buttons

Checkboxes are used to to create "radio buttons", that is, mutually exclusive checkboxes. Radio buttons are called this because like the car buttons on a radio, only one can be selected at a time.



>To create radio buttons using the checkbox

Follow the same steps as above but be certain to

1. Turn off the borders of each checkbox
2. Select Wingdings as the fill font
3. Select the empty circle and filled circle with a dot as the fill characters for off and on
4. Select Tab/Accessibility Order
5. Ensure that the radio buttons are in uninterrupted tabbing order and are consecutive
6. Highlight the radio button fields that are mutually exclusive
7. Click Select Fill Group

Buttons

Buttons can execute a script or a macro.

Scripts are form-level functions. Scripts are executed internally by the form. Scripts, just like macros, are triggered by a user-generated event such as a Double-Click. Scripts can be built into the form by using Edit/Calculation.

Macros are application-level functions. Forms do not execute macros, instead user-generated events such as Click are forwarded to the application to execute the proper macro. Macros are built into the application using the development language used by the application. Macros can be developed in C++, Visual Basic, JavaScripts, VB Scripts, Delphi, and many more.

>To run a script with the button object

1. Open the Properties window.
2. Under Edit/Calculation, write an action script.

Example: ALERT ("Routing Information: Send to district clerk who will match up approval with drawings and send to District Manager.\r\nApproval Information: District manager approval is required.")

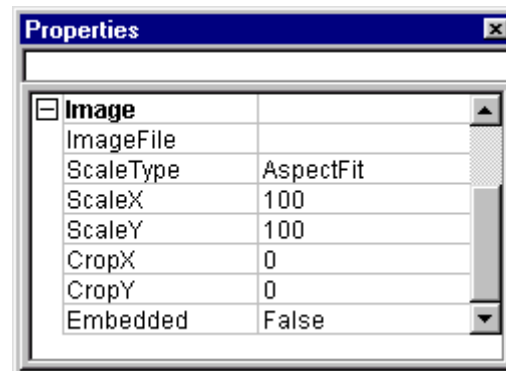
Note the use of " \r\n" within the text to force the information to a new line as returns are not recognized within the script.

>To trigger an event using a button object

1. Open the **Properties** window
2. Use the button object name within your application coding
3. Under the **Notify** heading, double-click the appropriate **Notify Flag** to execute the macro.
 - Click executes the macro when the user clicks the button.
 - DbClick executes the macro when the user double-clicks the button.

Images

Use the **Image** object to add images and logos to forms.



>To define the properties of an image

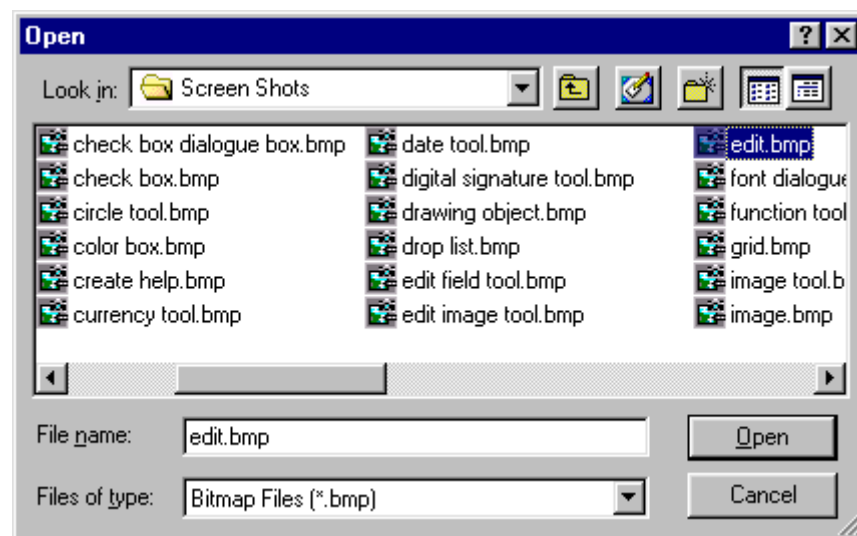
1. Open the **Properties** window.
2. Under the **Image** heading, click the drop-down menu in the ImageFile field.

The **Open File** dialog box appears.

3. Search to find the image in the file directory.



Supported image formats are: BMP, WMF, TIFF, JPEG, PCX, EPS, PNG.



4. Select the file and click the **Open** button.

The image appears on the form.

5. In the **ScaleType** field, select one of the following from the drop-down menu:

- **Aspect Fit**

This selection maintains the original proportions of the image file, and the image is sized to fit the outline of the box drawn with the object tool.

- **Fit In Box**

Regardless of the original proportions of the image file, this selection resizes the image to fit the image outline of the box drawn with the object tool.

- **By Factor**

not operable at this time

6. The functions for ScaleX, ScaleY, CropX and CropY are inoperable at this time.

7. Click on the drop-down menu in the **Embedded** field to select an option.

Select **True** to embed the image into the form. This places a copy of the image into the form and becomes a part of the form.

Select **False** to reference the image as a separate file. This selection calls the image file from a hard drive or a server. Only the pointer to the image is maintained as a part of the form; the actual image is not part of the form.

If using the Visual eMerge product, **False** is the correct selection. Using this method makes it easier to change images across a large number of forms at one time without changing each individual form.

Editable Images

Editable images allow insertion of pictures, logos, sketches, images of signatures, product pictures, etc. into a form while in the **Filler**. These images can come from anywhere, e.g., web, user's hard disk, floppy, server.

After drawing the editable image outline, you can define the properties (Edit, Borders, Margins and Text) just as you did when drawing **Fill** objects.

>To define the properties of an editable image

1. Open the **Properties** window.
2. Under the **Image** heading, click the drop-down menu in the **ImageFile** field.

The **Open File** dialog box appears.

3. Search to find the image in the file directory.
4. Select the **ImageFile** and click the **Open** button.
5. Define the remaining values just as you would for an Image object.

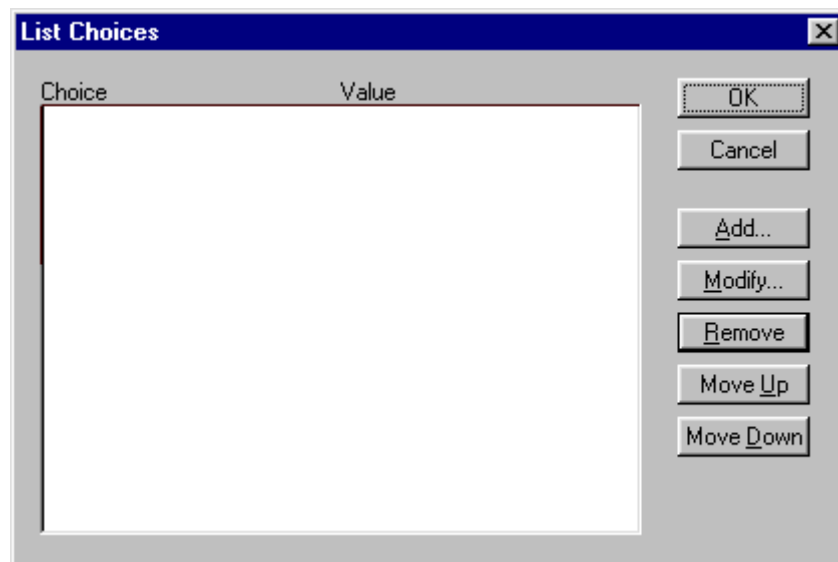
Drop Lists

A drop list defines the correct choices for the end user. When used in its original format, this restriction assures that the answer given will always fit the parameters and is self validating.

In its original format, the drop list is closed so that the only choices available are the only choices shown. Or, the drop list can be made an editable field. In this format, the user can add an answer that is not predefined in the drop list.

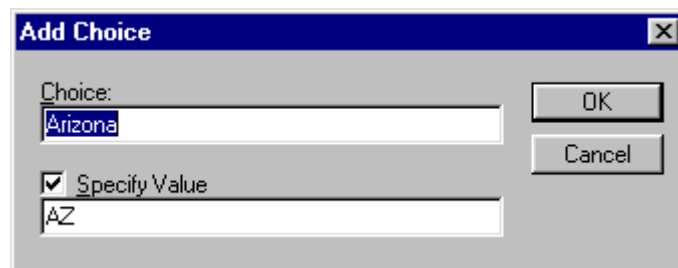
1. Open the **Properties** window.
2. Under the **Appearance** heading, enter a value in the **ListWidth** field. This value refers to the width of the drop-down menu in the **Filler**. You can enter a value, or let the width be determined by the text you enter.
3. Under the **Appearance** heading, enter a value in the **ListHeight** field. This value refers to the height of the drop-down menu in the **Filler**. You can enter a value, or let the height be determined by the text you enter.
4. Click the drop-down menu in the **List** field.

The List Choices dialog box appears.



5. Click the **Add** button.

The **Add Choice** dialog box appears.



6. Enter the information in the **Choice** field. Choice represents what the form user will see.

In this example, US states are used.

7. Check the **Specify Value** box, then enter the value in the field below. Value represents the actual text stored in the database or the information displayed on the screen when the drop down dialog is closed.

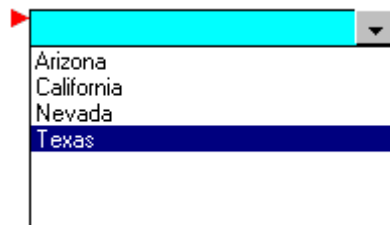
By doing so, the drop-down menu will offers a choice of states. When the user selects a state from the list, the value returned will be the two-letter abbreviation.

8. Click the **OK** button.
9. Click the **Add** button to enter another choice.
10. Repeat until all of the choices have been added.
11. Click the **OK** button.

These are the choices that the user has in the **Filler**.



Width and Height properties control the size of the Drop List field on the form. ListWidth and ListHeight properties control the size of the selection window that appears when users click the drop-down menu.



This is the value that is returned when the user selects **California**.



>To create an editable drop list

The drop list can become a an editable drop list in which the user can add his own information.

1. Open the Properties window.
2. Under the **Edit** heading, change User Modify to True.

Tables

Table is a collection of objects that are arranged and aligned neatly in a grid.


The main purpose of the table object is to maintain objects within it.

Note: Cells in a column of a table are of the same type of object.

>To create a table



If you have imported a form from FormFlow, the scollable tables feature is not supported.

1. Before you add a table to your form you should know the answer to the following:
 - Number of columns in the table
 - Number of rows in the table
 - What is the width of each field?
 - What type of objects should each column be (Edit, Text, Number, CheckBox, etc.)
 - What names you want to give to each column
 - Should the table manage the header for columns? If so, what should the height of the column header be?
 - Should the table manage the header for the rows? If so, what should the width of the row header be?
2. Locate the table icon  on the Object Bar and click on it
3. Position, size and place the table object on to your form as you would with any other object. The table dialogbox appears.
4. Set table properties
 - Select the number of rows and columns
 - Set the object type for each column
 - Give each column a name
 - Set the field width
 - Set the column header and the height of the header
 - Set the row header and the width of the header
5. Click Ok



Create the table and define the field properties without the header. Once this is completed, then add the header. In this manner, you don't have to go back and change the properties in the fields that make up the header row.

#	Item Number	Description	Qty	Cost (ea.)	Extended Cost	T
1	12300	Computer Desk	2	\$350.00	\$700.00	✓
2	3456	Printer	12	\$1.20	\$14.40	✓

>Accessing objects in a table

Access to objects within a table is available by:

1. Selecting a single cell

- Select the table
- Click on to the cell you want to select. The cell will be selected

2. Selecting all cells in a single column

- Select the table
- Move the cursor to the upper most edge of the table, cursor will turn into

an arrow pointing down

- Click, and the column pointed to by the arrow will be selected

3. Selecting all cells in a single row

- Select the table
- Move the cursor to the left most edge of the table, cursor will turn into an arrow pointing right
- Click, and the row pointed to by the arrow will be selected

4. Selecting all cells in the table

- Select the table
- Hold [SHIFT] and click on any of the cells. All cells within the table are selected.

>Resizing tables

You can resize the table in many ways:

1. Resize the entire table object, which will resize all cells proportionally

- Select the table
- Click and drag on the blue handles to resize the table

2. Resize a single column, which will resize all cells within the column

- Select the table
- Position your cursor over the dividing line between two columns. Watch the cursor change to two parallel lines and two arrows. Click and drag to right or left to resize the column width.

- or -

- Select the table
- Select the column number and enter a new value in Field Width

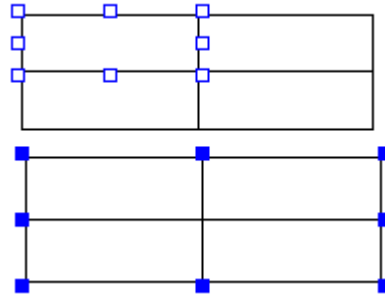
3. Resize a single row which will resize all cells within the row.

- Select the table
- Position your cursor over the dividing line between two rows. Watch the cursor change to two parallel lines and two arrows. Click and drag to top or bottom to resize the row height.

>To change the properties of a table cell

1. Click a table once to select it.

2. Click again to select an individual table cell.



-or-

3. Hold [SHIFT] and click a table cell to select all of the table cells.
4. Change the properties of the table cell as described in “Changing Object Properties” on page 42.

>To change the position of a table

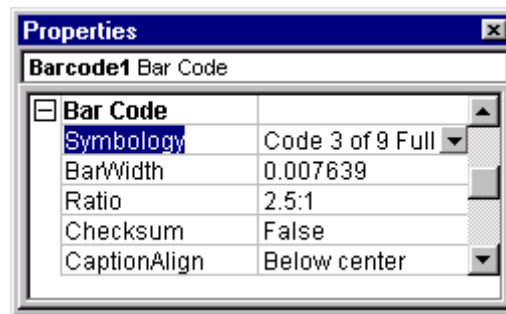
1. Right-click the table to display the table position **Properties** window.
2. Enter the **Left**, **Top**, **Width**, and **Height** values to change the position and size of the table.

- or -

1. Select the table
2. Drag and drop to the new location

Bar Codes

Bar codes consist of a series of vertical bars of varying widths, in which each of the digits zero through nine are represented by a different pattern of bars. They are commonly found on consumer products and are used especially for inventory control. When you first create a bar code object, Visual eForms Designer uses the Code 3 of 9 bar code format. This bar code is intended for placement only, and is overwritten as soon as you specify your own bar code symbology.



>To change the properties of a bar code object

1. Open the **Properties** window.
2. Under the **Bar Code** heading, click the drop-down menu in the **Symbology** field to select an option.

The bars and spaces in each symbol are grouped in such a way to represent a specific ASCII character or function. The interpretation of these groups is based on particular sets of rules called symbologies. See the following page for a list of bar code symbologies.

3. Enter a value in the **Bar Width** field.

This is a value indicating the width of the narrowest bar, specified in inches.

4. Click the drop-down menu in the **Ratio** field to select an option.

The **Ratio** sets a value that specifies the wide-to-narrow bar ratio.

5. Click the drop-down menu in the **Checksum** field to select an option.

This sets a value indicating whether to add a check digit to the bar code.

6. Click the drop-down menu in the **CaptionAlign** field to select an option.

This sets a value indicating the location of the caption characters relative to the bar code (Off, Below Left, Below Center, Below Right, Above Left, Above Center, or Above Right).

Bar Code Symbolologies

- Code 3 of 9 Full



- Interleave 2 of 5
- Codabar
- MSI
- Code 93, Full ASCII Set
- UPC A
- UPC E
- PostNet



- EAN/JAN 88
- EAN/JAN 13
- Code 128
- Standard Code 3 of 9
- PDF 417



Note: *Edit/Default Value Property is used for bar codes with pre-defined values.*

If Default Value is empty, the value can be entered by the user or come from the form scripts or application macros.

If you select a value in conflict with the Symbology, this message appears: Barcode value contains invalid characters for current symbology.

Digital Signatures

A digital signature is used when you want to distribute a form and enable the recipient(s) to authenticate your identity (the signer of the form). It can also be used to ensure that the original content of the form is unchanged. Additional benefits of the use of a digital signature are that it is easily transportable, can be time-stamped, cannot be easily repudiated or imitated by someone else.

To ensure form and data integrity, use several features of Visual e-Forms in combination. Protect the form design by using the password feature in Form Setup. Protect the data and the form by using the digital signature. The digital signature (1) creates a 128-bit encrypted fingerprint (or hash) of the form and (2) links the data in the form fields to the form's encrypted fingerprint.

The digital signature is invalid if the form is changed. The digital signature is invalid if the data is changed.

Several types of digital signatures are supported:

- NT Domain - Digital signatures are created using the login-name/password information of the user. The NT Domain server holds the login-name/password and acts as a Certificate Authority..
- Hand signatures - Digital Signatures are created using the actual hand signature of the user. This is drawn using a regular mouse or a pen mouse.
- PKI/Verisign - Digital signatures are created using a public-key signature algorithm. An example of this is the RSA public-key cipher. Any public-key Certificate Authority supported by the Microsoft CryptoAPI such as Verisign can be used with PKI digital signatures.
- Entrust - Entrust digital signature verification is based on Entrust X.509 version 3 certificates. Cerenade is an Entrust Partner.
- TOPAZ - Digital signatures are created using a signature pad from TOPAZ.

REQUESTOR	APPROVAL (DELEGATED)
 Signature Valid	Phillip Y. <small>O="VeriSign, Inc.", OU= www.verisign.com/ Ref, LIA8.LTD@98", OU OU=Digital ID Class 1 - CN=Phillip Y., E=phillip@ on 2/6/2004 10:18:36 AM</small>

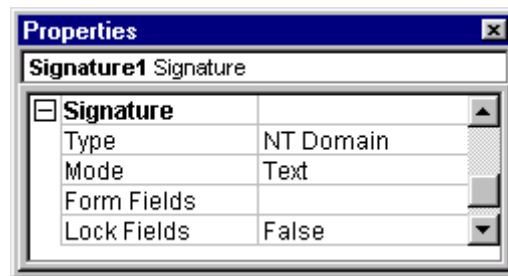
REQUESTOR	APPROVAL (DELEGATED)
 Signature Invalid	*Invalid*

Enhanced Digital Signature Support

>The form signing process

A one-way hash function is a means of squeezing messages into a short digest that preserves data integrity like a fingerprint.

1. A one-way hash of the document is produced.
2. The hash is encrypted with the:
 - user's password for NT Domain signatures, or
 - internal representation of the Hand Signature, or
 - user's private key in case of PKI digital signatures.
3. A copy of the original document plus the encrypted and signed hash are transmitted.
4. The recipient produces a one-way hash of the document.
5. Using the digital signature algorithm, the recipient decrypts the signed hash with the sender's encrypted information.
6. If the signed hash matches the recipient's hash, the signature is valid and the document is intact.



Signature1 Signature	
<input type="checkbox"/> Signature	
Type	NT Domain
Mode	Text
Form Fields	
Lock Fields	False

>To draw a digital signature

1. Click the **Digital Signature** object on the **Object** toolbar.
2. Move the cursor into the form window where you would like to place the **Digital Signature** object.
3. Hold down the mouse button and drag to draw the object on the form.
4. When the object is the size you want, release the mouse.

The outline of the **Digital Signature** object is displayed in the form window.

>To associate form fields to a digital signature object

Your electronic signature approves data contained in certain fields on the form. Therefore, you need to specify which fields on the form your signature will approve. To do this you must assign one or more fields to the signature field.



On your form you can have as many signature fields as you want. Each signature must have at least one form field assigned to it.

1. Open the **Properties** window.
2. Under the **Signature** heading, click the drop-down menu in the **Type** field to select an option.
3. Click the **Mode** drop-down menu to select an option.

Available options are **Text** and **Enhanced**.

- If **Text** is selected, the signature field places the name of the signer into the signature field when the field is signed.
- If **Enhanced** is selected, the signature field places a signature certificate into the signature field when the field is signed.



Enhanced mode produces a Signature Certificate

Rich Text Objects

Rich text objects allow the user while in the **Filler** to change Fill Font attributes for one or more characters in the field. This object type should be used sparingly as it adds a large amount of overhead to XML data.

In contrast, an edit field allows the user while in the **Filler** to change the Fill Font attributes for the entire field only.

Editing Text in Forms

Text may be entered only into text objects, fillable text objects, or other fillable objects such as tables. Text cannot be entered into image objects, bar code objects, or shapes such as lines, circles, or boxes.

Your current position within the text is indicated by a flashing, vertical text cursor.

>To enter or edit text within an object

Select the **Edit** icon.

-or-

Double-click the object that you want to edit.

When selected for text entry or editing, the object is surrounded by a red dashed line.

>To select a portion of text

Just like in most word processing programs, you can select a portion of text within an object for editing.

1. Select the Edit icon
2. Move the pointer to the text you want to select.
3. Hold down your mouse button and drag the pointer over the entire portion of text.

As you drag the pointer, the text you select becomes highlighted, appearing as white text on a dark background.

Check Spelling

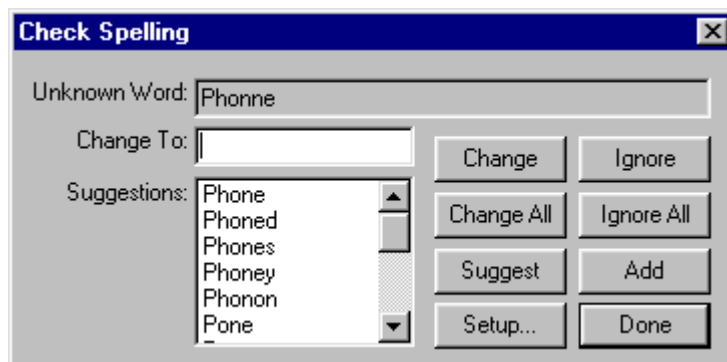
Use the Check Spelling feature to ensure that all text on your form is spelled correctly. You can check the entire form, including field help and default data entered in Preview Mode, or individual objects.

>To check the spelling of your document

Click the **Spell Check** icon on the **Standard** toolbar.

-or-

On the **Edit** menu, click **Check Spelling**.

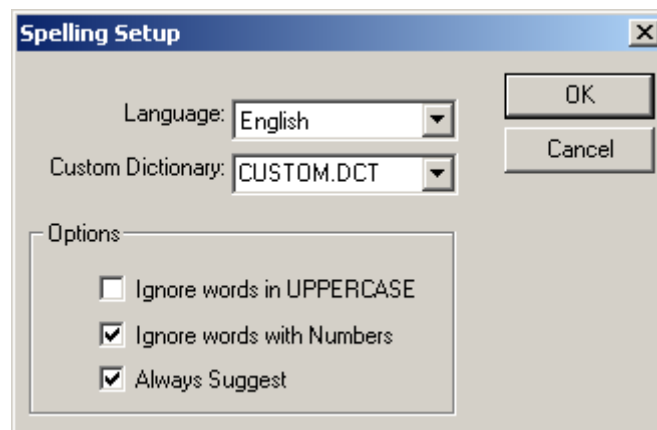


The **Check Spelling** dialog box appears with the first questionable word in the **Unknown Word** field. The suggested spelling, if there is one, appears in the **Change To** field. Other suggested spellings appear in the **Suggestions** list box.

>To change your spell checking options

1. Click the **Setup** button in the **Check Spelling** dialog box.

The **Spelling Setup** dialog box appears.



2. Click the **Language** and **Custom Dictionary** drop-down menus to specify your preferences for checking the spelling of your form.
3. Click the **OK** button.

>To add custom words to your dictionary

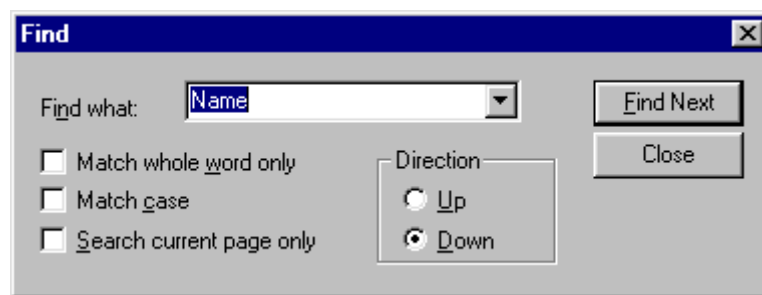
1. Click Setup. Select Custom.dct as the default.
2. If a questionable word is correct and you use it often, click the Add button to add it to the Custom.dct.

Find and Replace

The find and replace text function allows you to locate text within your current displayed form and replace it with other text.

>To find text

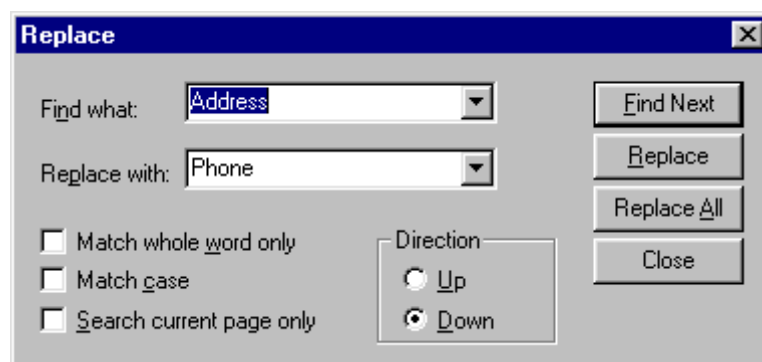
1. On the **Edit** menu, click **Find**. Enter the desired text in the **Find** dialog box.



2. If desired, check the necessary **Match** and **Search** boxes to narrow the search. In the **Direction** section, choose to search **Up** or **Down**.
3. Click the **Find Next** button.

>To find and replace text

1. On the **Edit** menu, click **Replace**. Enter the desired text in the **Find** dialog box.



2. Enter the replacement text in the **Replace with** field.
3. If desired, check the necessary **Match** and **Search** boxes to narrow the search. In the **Direction** section, choose to search **Up** or **Down**.
4. Click on **Find Next**.

5. When the text is found, do one of the following:

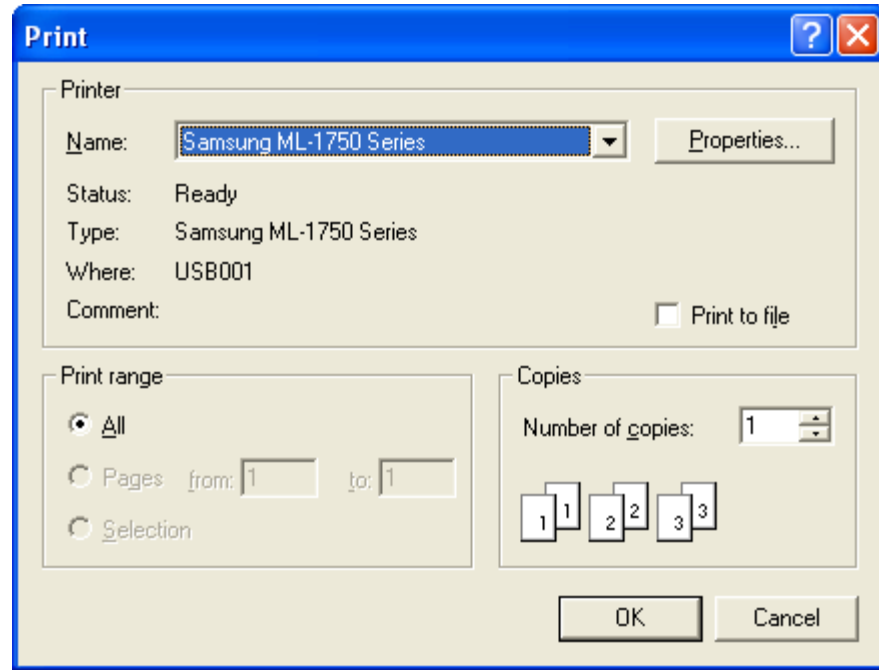
- Click **Replace** to replace the given text occurrence.
- Click **Replace All** to replace all text occurrences in the form.

Printing Forms

>To print a form

1. On the **File** menu, click **Print**.

The **Print** dialog box appears.



2. In the **Printer** section, select a printer from the drop-down menu.
3. In the **Print range** section, select pages to print.
4. To print to a file, check the **Print to file** box.
5. In the **Copies** section, select the number of copies to print.
6. Click the **OK** button.

Previewing Forms

When you have finished creating your form, you can test it in **Preview** mode to see how it will appear to users.

>To go to preview mode

1. On the **File** menu, click **Preview**.

- or -

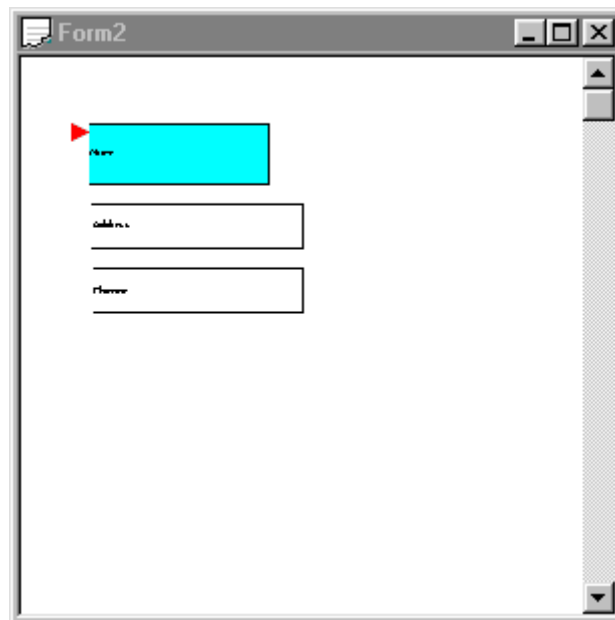
Click the Preview Mode icon  on the Standard Toolbar

- or -

press Cntrl+F2

Your current position is noted by a turquoise field and red arrow.

Data on the form during Preview Mode becomes the default data in each object. Be certain to clear this data prior to your final save. To clear the form in Preview mode, press Control+E. The system asks for confirmation before clearing.



2. Enter the desired data in the field.

After entering data in the first field, proceed to the remaining fields.

To...	Press...
go to next field	[TAB]
go to previous field	[SHIFT] + [TAB]
go to next line in field	[ENTER]
clear data in the form	[CNTRL] + E

Sticky Notes

Text can be added to the form while in Preview mode. This is helpful to users during testing.

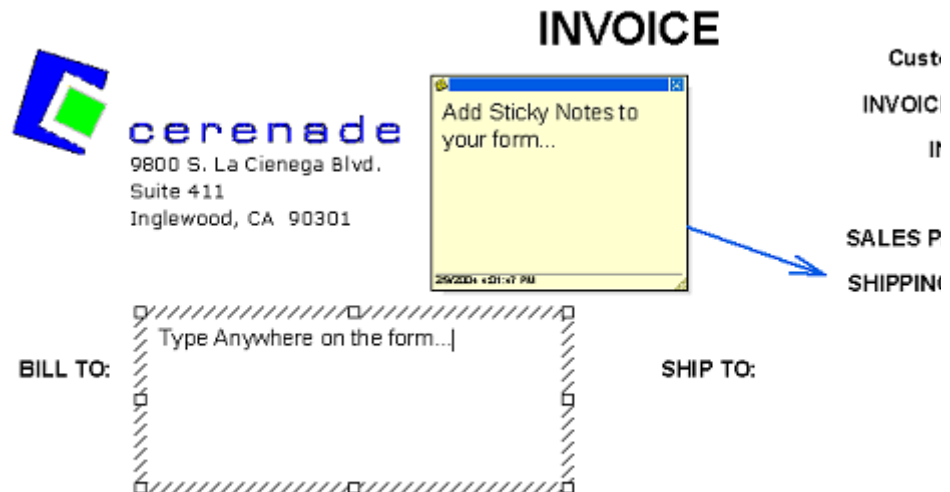
>To add sticky notes

1. While in Preview mode, press Control+N
2. The cursor changes to sticky note icon
3. Enter text

Any text added in this manner will be included in the database information as an XML string.

>To see the options for this feature, right click on the sticky note

1. Iconic (show the note in its entirety or as an icon)
2. Transparent (selected or not selected)
3. Printable (selected or not selected).
4. Callout (associate the note to text using an arrow)
5. Delete (remove the note)

**Type Anywhere**

Text can be added to the form while in Preview mode. This is helpful to users when additional text is required and there is no fill in space made available or the fill in space is too small.

>To add text using Type Anywhere

1. While in Preview mode, press Control+shift+ N

2. The cursor changes to a text Type Anywhere icon.
3. Enter text.

Any text added in this manner will be included in the database information as an XML string.

>To see the options for this feature, right click on the Type Anywhere border.

1. Transparent (selected or not selected)
2. Printable (selected or not selected)
3. Callout (associate the note to text using an arrow)
4. Delete (remove the note)

Exiting Visual eForms Designer

>To exit Visual eForms Designer, do either of the following

On the **File** menu, click **Exit**.

-or-

Click the **X** in the top right corner of the screen.

Accessibility

Visual eForms Designer meets all MSAA requirements with its own assistive accessibility feature. This feature complies with Section 508 regulations, mandating accessibility for visually-impaired users. **No additional components or third party plug-ins are required** for Designer's accessibility features. All forms created in designer are instantly compatible with all audio-enabled assistive technology systems with support for MSAA.

Every fillable object within a form can be created easily with accompanying computer-generated audio to meet the needs of visually-impaired users.

For true accessibility, all objects on the form must be heard. Thus, instructions, form title, form number, section headings and so forth all should be read to the user. All images must be identified as well, for example, "The seal of the State of Texas."

Making Forms Accessible

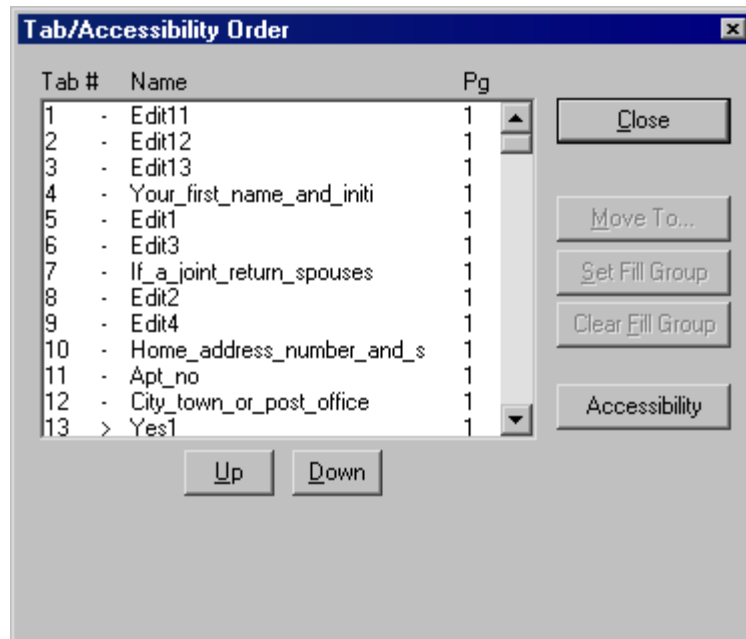
Visual eForms Designer simplifies the process of making forms accessible. The general process of making a form accessible involves associating accessibility-text with each and every fillable field on the form. Once a field on the form gets the filler's focus, its accessibility-text will be verbalized to the visually-impaired user. While this process is simple in concept, if features in the Designer providing support for creating Field to accessibility-text association are not carefully designed, making a form accessible can be a lengthy, cumbersome and painful task to undertake.

>To create a Field to accessibility-text association

With Visual eForms Designer, the **Tab/Accessibility Order** dialog box is used to create all Field to accessibility-text associations on the form.

1. On the **Objects** menu, click **Tab/Accessibility Order**.

The Tab/Accessibility Order dialog box appears:

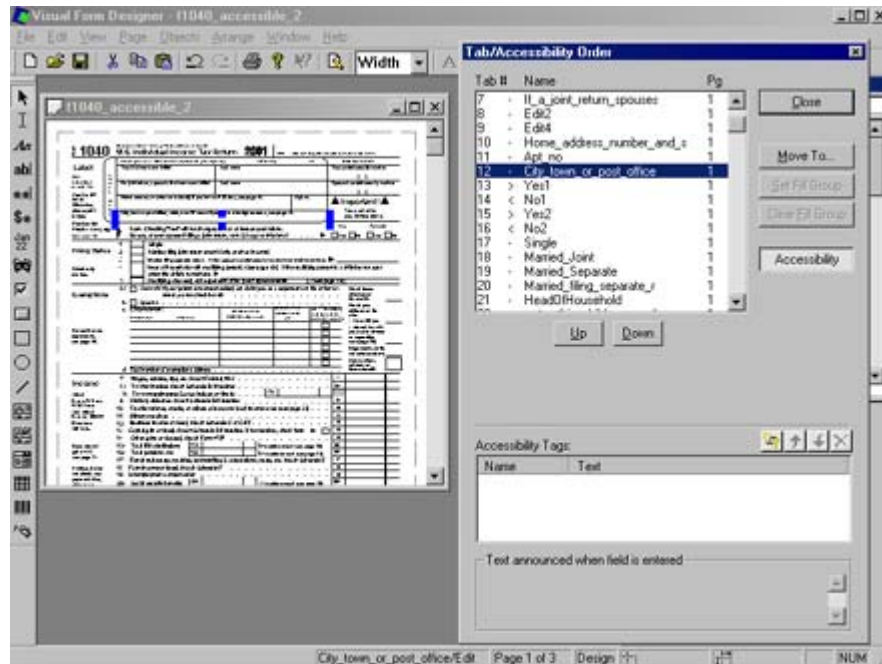


2. Click the **Accessibility** button to enable Accessibility mode.
3. In the dialog box list, select the field for which you want to set up accessibility text.

The text field is highlighted in the list AND on the form itself.

- On your form, locate the selected field for which you want to set up accessibility text.

Move the dialog box if you cannot see the entire form. Selecting a field in the dialog box also selects the field on the form, as shown below:



- Drag the field(s) from the form and drop it into the **Accessibility Tags** section of the dialog box.

The name of the field(s) appears in the Accessibility Tags section of the dialog box.

- Drag and drop one or more text fields from the form into the **Accessibility Tags** section of the dialog box.

The text appears in the Accessibility Tags section of the dialog box.

Note: If the text changes in the form, Designer will automatically update all tags using the text.

- If you wish to include the field's Help text as part of the text announced when the field is entered, click the Field Help icon.



This adds <FieldHelp> to the list of current Accessibility Tags for the Field.

8. If you need to change the order of the tags, use the **Up** and **Down** buttons to move tags up or down. Use the **Delete** icon to delete a tag from the list of Accessibility Tags. With any change made to the Accessibility Tags, note the **Text announced when field is entered** section.
9. Repeat steps 3 through 7 for all of the fields on the form.

HINTS TO IMPROVE ACCESSIBILITY DESIGN

>For items such as instructions, section headings, form title or number, which must be read but not filled in, simply drag and drop them to the Accessibility Tag section of the field before or after the item.

>For formatted fields such as Dates or Mask, you need to add text to FieldHelp property of the field in order to describe the typing format to use. Make sure FieldHelp is added to the Accessibility Tag section of the field.

>Accessibility reads in this sequence: accessibility tags, field type such as “editable text”, state the field is in such as “Checked or unchecked for checkbox fields.”

>Drop lists are handled automatically by Visual eForms.

>You can drag and drop the same object (text, edit, etc.) into Accessibility Tags as often as you need to.

>Tables are handled automatically by Designer which treats the text from column headers of the Table as Accessibility Tags. Row numbers are also handled automatically. There is no need to add them to the Accessibility Tags section.

>As you make changes to the text on the form, caption on a field or the field help, this same information is automatically updated in the Accessibility Tag.

>Select the items in the order you want them read so that the list within Accessibility Tags does not have to be resequenced.

>To test your form, you need to have accessibility software installed and running. Then, switch the Visual eForms Designer from Design Mode to Preview Mode and test.

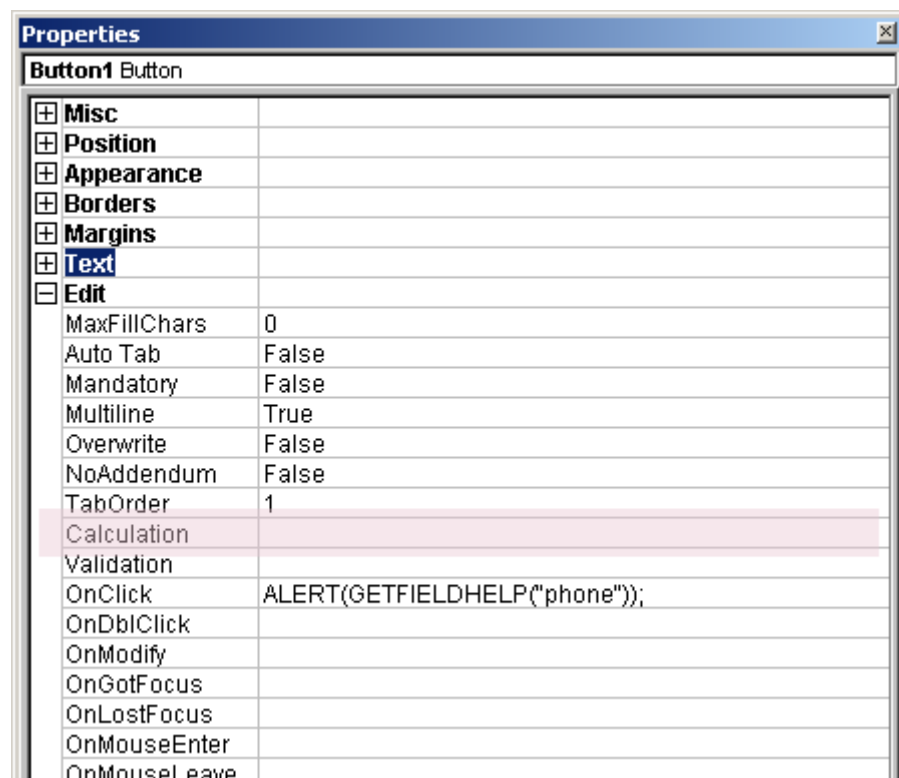
Note: On windows NT, 2000 and XP you can test accessibility using Microsoft Narrator, a FREE screen reader.
To access Narrator click on Start, Programs, Accessories, Accessibility, Narrator

Scripts

Scripts build intelligence into forms. They enable users to create powerful, customized forms. Visual eForms Designer supports three types of Scripts:

1. **Calculations Scripts** - These scripts are used to calculate a field dynamically based on values of other fields on the form. Changing the value of any of the fields with which the calculation is built will automatically recalculate the value of the field with the Calculation Script. For example, the form designer can set the value of field A to be calculated automatically by adding the value of field B to value of field C and dividing the result by 3 with the following Calculation Script:

$$([B] + [C]) / 3$$



2. **Validation Scripts** - These scripts provide a mechanism to validate the value of a field whenever the value is changed. For example, the form designer can enforce the changes to value of field A to be accepted/validated only if the new value falls between integers 5 and 10 inclusively with the following Validation Script:

$$(5 \leq [A]) \text{ and } ([A] \leq 10)$$

Overwrite	False
NoAddendum	False
TabOrder	1
Calculation	
Validation	
OnClick	ALERT(GETFIELDHELP("phone"));
OnDbClick	
OnModify	
OnGotFocus	
OnLostFocus	
OnMouseEnter	
OnMouseLeave	
FieldHelp	

3. **Action Scripts** - These scripts/handlers allow the form designer to specify actions taken upon triggering of events. Visual eForms Designer supports the following Action Scripts/handlers: OnClick, OnDbClick, OnModify, OnGotFocus, OnLostFocus, OnMouseEnter and OnMouseLeave. For example, the form designer can effect the display of a "Hello, World!" message whenever field A is clicked with the following OnClick Handler:

```
Alert("Hello, World!")
```

Properties	
Button1 Button	
+	Misc
+	Position
+	Appearance
+	Borders
+	Margins
+	Text
-	Edit
	MaxFillChars 0
	Auto Tab False
	Mandatory False
	Multiline True
	Overwrite False
	NoAddendum False
	TabOrder 1
	Calculation
	Validation
	OnClick ALERT(GETFIELDHELP("phone"));
	OnDbClick
	OnModify
	OnGotFocus
	OnLostFocus
	OnMouseEnter
	OnMouseLeave
	FieldHelp
	FillChar None
	Default Value

Details of Scripts

Each script has three parts:

- The *function* determines what the script does, such as **ROUND** or **NUM**.

```
NUM([Edit1])+NUM([Edit2])+NUM([Edit3])
```

- The *expression* determines the values or fields used in the script.

```
NUM([Edit1])+NUM([Edit2])+NUM([Edit3])
```

- An operator determines how expressions and functions interact with each other.

```
NUM([Edit1])+ NUM([Edit2])+ NUM([Edit3])
```

Functions

A function is a formula for a specific kind of script. The functions make it quicker and easier to create scripts.

For example, if you need to add a column of numbers like the one below, you could write a script like this:

```
[Cost:1]+[Cost:2]+[Cost:3]+[Cost:4]+[Cost:5]+[Cost:6]
```

Plants	Quantity	Cost
Hawaiian Palms	2.00	\$9.00
Banana Palms	1.00	\$11.00
Orchids	3.00	\$13.00
Ferns	2.00	\$12.00
Hibiscus	0.00	\$0.00
Other	4.00	\$9.00

▶ \$54.00

Rather than enter a lengthy script, you can use the **SUM** function to achieve the same result:

```
SUM([Cost])
```

The **SUM** function is only one of forty-one built into Visual eForms Designer.

Built-in Functions

All forty-one functions are listed below, but not every function is available in Calculation, Validation or the Action Scripts such as OnClick, OnDbClick, etc.

Function	Description
ALERT	Displays a message box containing the specified text.
CLEARDATA	Clears the Default property of all fields.
DATE	Returns today's date in MM/DD/YYYY format.
DAY	Returns the DD portion of a date in MM/DD/YYYY.
DIFFDATE	Returns the number of days between two date expressions.
DIFFTIME	Returns the number of hours between the two time expressions.
GETCURRFIELD	Returns the current field in focus.
GETCURRPAGE	Returns the current page number.
GETFIELDHELP	Returns the help text of the specified field.
GETFIELDPROPERTY	Returns the value for the requested property of the specified field.
GETNUMPAGES	Returns the number of pages on the form.
GETUNFILLEDMANDATORY	Returns the name of the first field on the form that has been marked Mandatory and has not been filled with any value.
GOTOFIELD	Moves focus to the specified field.
GOTOPAGE	Displays the specified page.
HOUR	Returns the HH portion of time in HH:MM:SS.
IF	Evaluates a condition and returns one of two different values, depending on whether the condition is met (true) or not met (false).
LEFT	Returns a specified number of characters starting from the left of the string.

Function	Description
LOWER	Returns the lower case representation of the string.
LTRIM	Removes the leading spaces from the left of a string.
MINUTE	Returns the MM portion of time in HH:MM:SS.
MONTH	Returns the MM portion of a date in MM/DD/YYYY.
NUM	Converts an expression into a numeric value.
PRINTDIALOG	Invokes the print dialog box in order to print the form
RIGHT	Returns a specified number of characters starting from the right of the string.
ROUND	Numerically rounds the expression. The number of decimal places is specified as part of the function.
RTRIM	Removes the leading spaces from the right of a string.
SEC	Returns the SS portion of time in HH:MM:SS.
SETFIELDDATA	Sets the content of the specified field.
SETFIELDPROPERTY	Sets a property of the specified field.
SETPROPERTY	Modifies properties of a field.
STR	Converts an expression into a string.
STRAT	Returns a string beginning at the specified position.
STREXTRACT	Returns a subset of a string.
STRINSTR	Searches and returns the string within a text.
STRLEN	Returns the length of a string.
SUM	Returns the sum of the cells of a table column.

Function	Description
SUMDATE	Returns a date that is specified number of days [i.e., <num exp>] before or after another date [i.e., <date exp>]
SUMTIME	Returns a time that is specified number of hours [i.e., <num exp>] before or after another time [i.e., <time exp>]
TIME	Returns the current time in HH:MM:SS format.
UPPPER	Returns the uppercase representation of the string.
YEAR	Returns the YY portion of a date in MM/DD/YYYY.

Expressions

An expression can be any of the following:

- A field that you want to reference in a script. In this instance, the expression is enclosed in parenthesis, and if there are two or more in a row, separated by commas.

```
( [ Edit1 ] )
```

```
( [ Number1 ] , [ Number2 ] )
```

- Data that you want to reference does not require parenthesis.

```
99 , 999 . 00
```

```
4
```

- A string that you want to reference does not require parenthesis.

```
"Address"
```

```
"12/25/2001"
```

Examples:

- "TOTAL SALES = " + NUM([TOTAL]) + NUM([TAX]) * 0.75 + " DOLLARS."
- ROUND([SALARY],2)
- STREXTRACT("CERENADE",0,2)
- IF([Cerenade_Check]>0,"http://www.cerenade.com",if ([eForms_Check]>0,"http://www.visualeforms.com",""))

Operators

Operators determine how expressions and functions interact with each other.

Operator	Explanation	Example
+	Addition	[Number1]+[Number2]
-	Subtraction	[Number1]-[Number2]
/	Division	[Number1]/[Number2]
*	Multiplication	[Number1]*[Number2]
%	Remainder or modulus	IF([Number3]>12,[Number3]%12,[Number3])
^	To the power of	[Number1]^4
=	Equal to	IF([Number1]=[Number2],1)
,	Expression separation	IF([Number1]=[Number2],1,0) <i>This is read as "If field number 1 equals field number 2, then yes, else no"</i>
:	Table cell specification	[Quantity:1]+[Quantity:2]
<	Less than	IF([Edit2]<4,"Red","Blue")
>	Greater than	IF([Edit2]>4,"Red","Blue")
<>	Not equal to	IF([Edit2]<>4,"Red","Blue")
<=	Less than or equal to	IF([Edit2]<=4,"Red","Blue")
>=	Greater than or equal to	IF([Edit2]>=4,"Red","Blue")
& or And	And	[Edit1] & [Edit2] [Edit1] And [Edit2]
or Or	Or	[Edit1] [Edit2] [Edit1] Or [Edit2]
~ or Not	Not	[Edit1] ~ [Edit2] [Edit1] Not [Edit2]

Note: Operators must be outside the expressions, separating expressions.

For example, these are *valid* scripts:

- [Number2]+[Number3]+[Number4]
- NUM([Number2]+[Number3]) + NUM("12"+"4")

This is an *invalid* script:

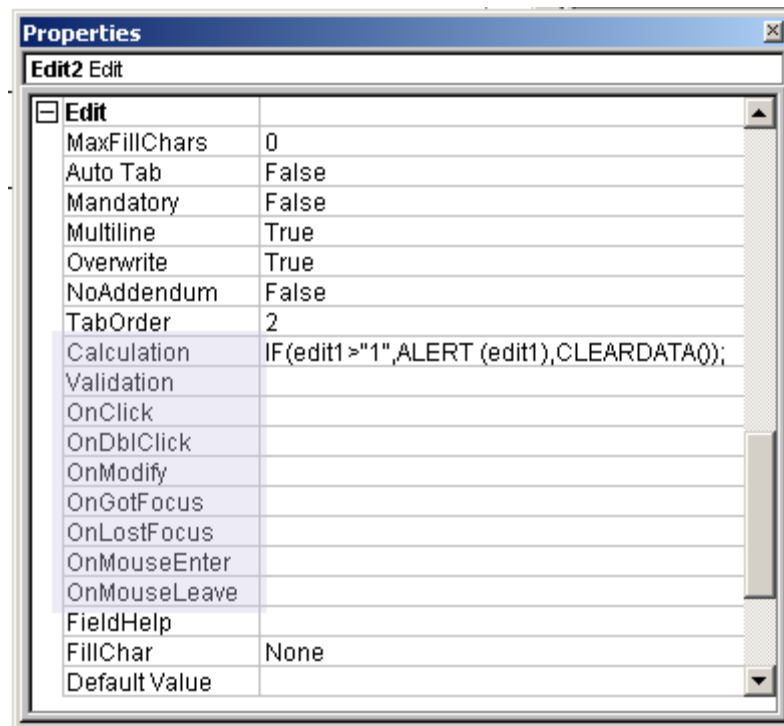
- [Number2]+[Number3+][Number4]

Creating the script

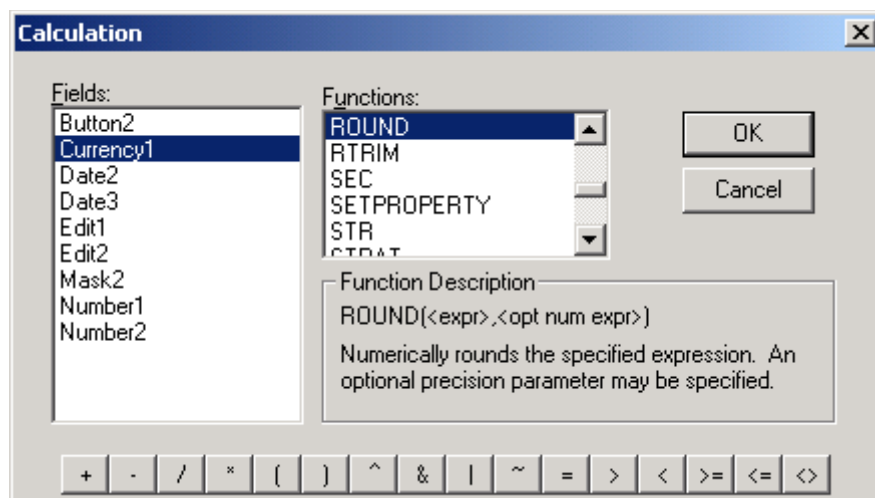
Creating a script for calculation, validation or any of the user-events (OnClick, OnDbClick, OnModify, etc.) requires the *function, expression and operator*.

>To create a script

1. Select a fillable field in the form and open the **Properties** window.
2. Under the **Edit** heading, click the drop-down menu in the Calculation, Validation, OnClick, OnDbClick, etc. field.



The appropriate dialog box appears. The name of the field that you have selected is *not* displayed in the dialog box.



3. Double-click a function name in the **Functions** section.

The **Function**, in this case **ROUND** and its syntax, appears in the calculation field.

4. Replace <expr> with [Currency1], as shown here:

```
ROUND([Currency1],<opt num expr>)
```

5. Replace <opt num expr> with 2, as shown here:

```
ROUND([Currency1],2)
```

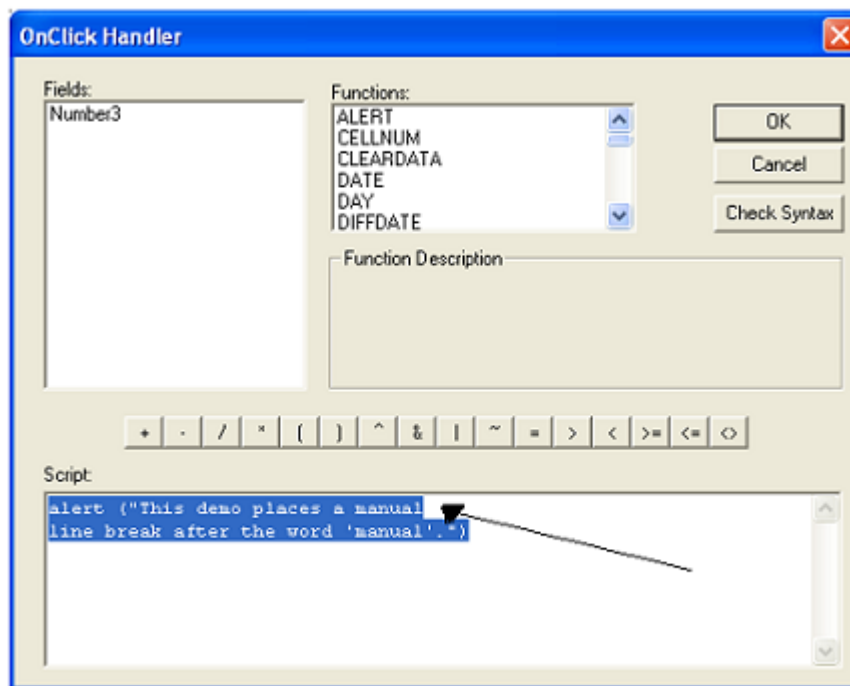
6. Click the **OK** button.

The script is placed in the field that you had selected and is executed in **Fill** mode.

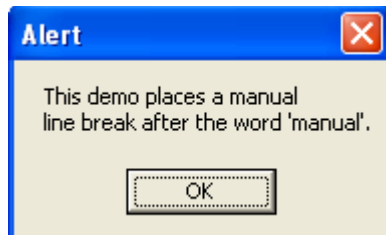
The action that will occur is the currency amount shown in the Currency1 field will be rounded to two decimal places.

>To force a line break in your calculation display

1. While in a Calculation, press [CNTL]+[Enter] at the point where you want to force a new line.



2. Go to Preview Mode to check the result.



Built-In Functions

The following is a list of all available functions in Visual eForms Designer, their explanations and syntax, as well as script examples.

Function: **ALERT**

Syntax: ALERT (Prompt)

Explanation: Displays a message in a dialog box, waits for the user to click a button.

Parameter: The following parameters are available

Parameter	Description
Prompt	String expression displayed as the message in the dialog box. The maximum length of <i>prompt</i> is approximately 1024 characters, depending on the width of the characters used. If <i>prompt</i> consists of more than one line, you can separate the lines using a carriage return character (\r), a linefeed character (\n), or carriage return–linefeed character combination (\r\n) between each line.

Notes: Since Alert is a function, it is not available in the list of available functions for Calculation scripting.

See also: None.

Example:

Enter: ALERT (“Signature is Verified.”)

Returns: Signature is Verified.

Enter: ALERT(“Please Enter Your Name\rYour Date of Birth\nand Phone\r\nThanks.”)

Returns: Please Enter Your Name
Your Date of Birth
and Phone
Thanks.

Function: **CLEARDATA**

Syntax: `CLEARDATA ()`

Explanation: Erases data stored in the Default property. Also, erases the data entered into the form and blanks out the form.

Notes: NONE

See also:

Example: Clear data in the form: `CLEARDATA()`

Where:

Enter:

Returns:

Function: **DATE**

Syntax: `DATE ()`

Explanation: Returns the current system date in MM/DD/YYYY format.

Notes:

- `DATE()` returns a string.
- `DATE()` is an automatic script. The value of the field will automatically be set to today's date.
- `DATE()` returns a string, and therefore the field where a call to `DATE()` takes place must be an EDIT object.
- `DATE()` returns zero if called from a NUMBER object.

See also: `TIME()`, `DAY()`, `MONTH()`, `YEAR()`, `HOURL()`, `MINUTE()`, `SEC()`

Example: Extract the month portion of today's date: `STREXTRACT(STR(DATE()),0,2)`
Display today's date: `ALERT(DATE())`

Where: Assume that today is November 18, 2004

Enter: `STREXTRACT(STR(DATE()),0,2)`

Returns: 11

Enter: `ALERT(DATE())`

Returns: The above script displays today's date.



Function: DAY

Syntax: DAY (DateExpr)

Explanation: Returns a whole number between 1 and 31, inclusive, representing the day of the month.

Parameter: The following parameters are available

Parameter	Description
DateExpr	any expression that can represent a date. If <i>date</i> contains Null, Null is returned. DateExpr is formatted as MM/DD/YYYY

Notes: DAY() function returns a numeric value

See also: DATE(), TIME(), MONTH(), YEAR(), HOUR(), MINUTE(), SEC()

Example

Where:

Enter: DAY ("03/22/2004")

Returns: 22

Enter: DAY (DATE ())

Returns: If today's date is April 28, 2004 the above script returns: 28

Function: **DIFFDATE**

Syntax: `DIFFDATE (DateExpr1 , DateExpr2)`

Explanation: Returns the number of days between two dates.

Parameter: The following parameters are available.

Parameter	Description
<code>DateExpr1</code>	Begin date - Date expression formatted as MM/DD/YYYY
<code>DateExpr2</code>	End date - Date expression formatted as MM/DD/YYYY

Notes: `DIFFDATE()` function returns a numeric value.
Sequence dates so that the earlier date is first and the most recent date is second.

See also: `DATE()`, `TIME()`, `DAY()`, `MONTH()`, `YEAR()`, `MINUTE()`, `SEC()`

Examples

Enter: `DIFFDATE ("02/14/1959" , "+02/14/2004")`

Returns: 12462

Enter: `DIFFDATE ("02/14/1959" , DATE ())`

Returns: If today's date is February 13, 2004, then the above script returns: 12461

Enter: `SETFIELDDATA("MyAge",STR(ROUND(DIFFDATE([MyBirthdate],DATE())/365.3,0)))`

Returns: If today's date is January 18, 2005 and `MyBirthdate` is February 27, 1915, then `MyAge` is 89, rounded according to the parameters set in the calculation.

Function: **DIFFTIME**

Syntax: `DIFFTIME(TimeExpr1, TimeExpr2)`

Explanation: Returns the number of hours between two time expressions

Parameter: The following parameters are available

Parameter	Description
TimeExpr1	Begin time - String formatted as HH:MM:SS
TimeExpr2	End time - String formatted as HH:MM:SS

Notes: DIFFTIME() function returns a numeric value, rounded up.

See also: DATE(), TIME(), DAY(), MONTH(), YEAR(), MINUTE(), SEC()

Examples

Where:

Enter: `DIFFTIME("07:12:55","15:55:55")`

Returns: 9

Enter: `DIFFTIME("07:12:55","15:35:55");`

Returns: 8

Function: GETCURRFIELD

Syntax: GETCURRFIELD()

Explanation: Returns the name of the current field in focus.

Notes:

See also: GETCURRPAGE()

Examples

Where: Add field “SSN” to your form
Set the OnGotFocus action of field “SSN” to ALERT (GETCURRFIELD())

Enter: Go to Fill Mode
Move your mouse over the “SSN” field

Returns: You will get an Alert box with the name of the current field that has focus.

Function: **GETCURRPAGE**

Description: Returns the current page number of the currently loaded form.

Syntax: `GETCURRPAGE()`

Parameters: None.

Remarks: `GETCURRPAGE()` returns a Numeric value.

Return Values: Current page number of the currently loaded form.

Example:

- Add a button to your form
- Set the OnClick action of this button to
`ALERT("Current Page No.: " + STR(GETCURRPAGE()))`
- Go to Fill Mode
- Click on the button

The above example uses the `STR()` function to convert the Numeric value returned by `GETCURRPAGE()` to a String value.

The String value is then prefixed with "Current Page No.: " before it is displayed.

Function: GETFIELDHELP

Syntax: GETFIELDHELP(FieldNameStr)

Explanation: Returns the FieldHelp property for a field on the form.

Parameter: The following parameters are available

Parameter	Description
FieldNameStr	Name of the field on the form

Notes: Field name is expressed as a string or an expression that returns a string.

Examples

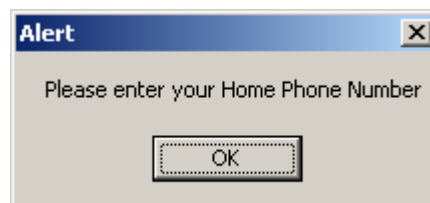
Where:

1. Add an edit field ("Phone") to your form
2. Add a button to your form
3. Set the FieldHelp property of "Phone" to
Please enter your Home Phone Number
4. Set the OnClick action of the button to
ALERT(GETFIELDHELP("phone"))

Enter:

- Go to Fill Mode
- Click the button

Returns:



Function: GETFIELDPROPERTY

Syntax: GETFIELDPROPERTY(FieldStr, PropertyIDStr)

Explanation: Returns the specific property of a field on the form.

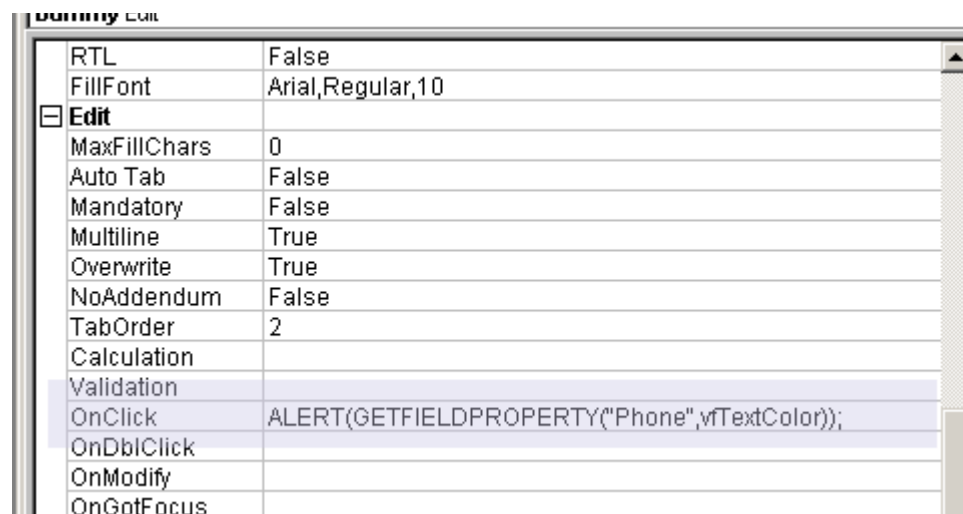
Parameter: The following parameters are available

Parameter	Description
FieldStr	Name of the field on the form
PropertyIDStr	One of the following properties: vfBackColor, vfBottomBorder, vfEnabled, vfLeftBorder, vfLineColor, vfMaxFillChar, vfPageNumber, vfRightBorder, vfRoundedBorder, vfTextColor, vfTopBorder, vfVisible

Notes: GETFIELDPROPERTY() function returns a string

See also: SETFIELDPROPERTY()

Examples Add 2 fields to your form. Name these fields: Phone and Dummy
Set the OnClick action of field 'Dummy' to
ALERT(GETFIELDPROPERTY("Phone",vfTextColor))



Enter: In the Fill Mode, click on field 'Phone'

Returns: The Alert screen will display the RGB value of the Text Color of field 'Phone'

To specify RGB for a Custom color, go to the appropriate color property instructions such as "To Change an Object's Background Color". On the color dialog box, note the numbers in the lower right hand corner for "red, green, blue." These are the numbers used to represent a custom color for parameter PropertyVal.

Function GETNUMPAGES**Syntax:** GETNUMPAGES ()**Explanation:** Returns the total number of pages for the currently loaded form.**Parameters:** None.**Remarks:** None.**Return Values:** GETNUMPAGES () returns a numeric value

Example:

- Add a button to your form
- Set the OnClick action of this button to
ALERT("This Form has " + STR(GETNUMPAGES()) + " pages.")
- Go to FillMode
- Click on the button

The above example uses the STR() function to convert the Numeric value returned by GETNUMPAGES() to a String value.

Function: GETUNFILLEDMANDATORY

Syntax: GETUNFILLEDMANDATORY ()

Explanation: Returns the name of the first mandatory field left blank on the form.

Notes: GETUNFILLEDMANDATORY() function returns a string

See also:

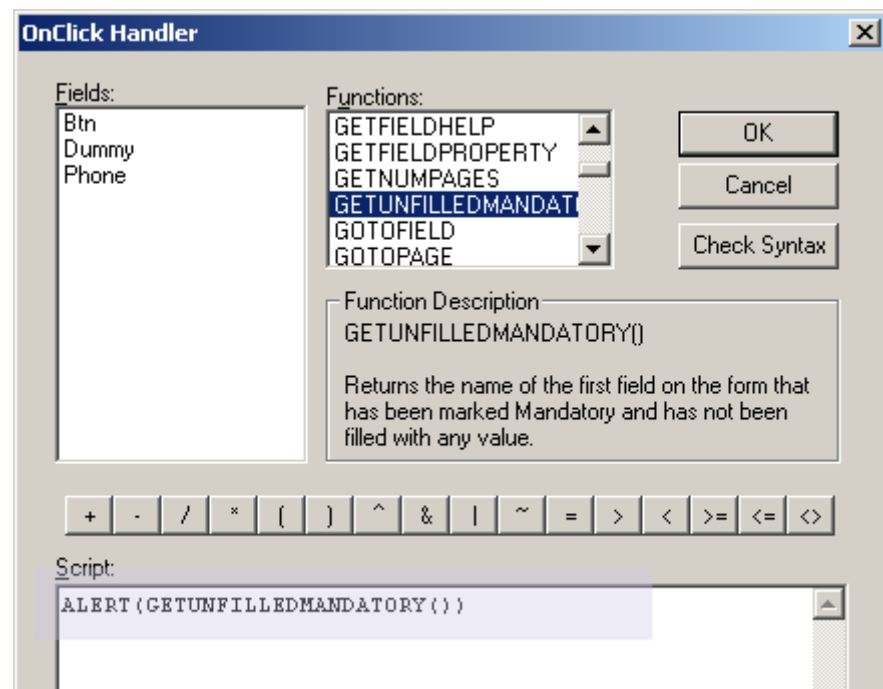
Examples

Add 2 Edit fields ('Phone' and 'Dummy') to your form

Add 1 Button ('Btn') to your form.

Set the 'Mandatory' property of field 'Phone' to TRUE.

Set the OnClick action of field 'Btn' to
ALERT(GETUNFILLEDMANDATORY())



Enter: In the Fill Mode, click on 'Btn'

Returns: The Alert screen will display the name of the first mandatory field that is left blank (i.e. 'Phone')

Function: **GOTOFIELD**

Syntax: GOTOFIELD (FieldName)

Explanation: Resets focus to field FieldName.

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the field on the form

Remarks: None.

Return Values: None.

Example: IF ([MALE]="1", GOTOFIELD("Q12"),GOTOFIELD("Q18"))

In the above example "MALE" is the name of a CheckBox field.

If "MALE" is checked, then cursor moves to field "Q12"; otherwise, the cursor will move to field "Q18".

Function: GOTOPAGE

Syntax: GOTOPAGE (PageNumber, NoScroll)

Explanation: Sets the current page of the form to PageNumber and sets the focus to the first field of the new page.

Parameter: The following parameters are available

Parameter	Description
PageNumber	Page we are switching to. PageNumber is a numeric value.
NoScroll	1: Do Not reset the scrollbar to the top of the page 0: Set scrollbar to the top of the page

Remarks: None.

Return Values: None.

Example: IF ([MALE]="1", GOTOPAGE(2,0), GOTOPAGE(3,0))

In the above example "MALE" is the name of a CheckBox field.

If "MALE" is checked, then move cursor to the first field of page 2; otherwise, the cursor will move to first field of page 3.

Function: **HOUR**

Syntax: `HOUR(TimeStr)`

Explanation: Returns the hour portion of the time.

Parameter: The following parameters are available

Parameter	Description
<code>TimeStr</code>	string expression in this format: HH:MM:SS

Notes: `HOUR()` function returns a numeric value..

See also: `DATE()`, `TIME()`, `DAY()`, `MONTH()`, `YEAR()`, `MINUTE()`, `SEC()`

Examples

Where: `[Edit1]` contains “10:15:22” and the current time is 9:25:22

Enter: `HOUR([Edit1])`

Returns: 10

Enter: `HOUR(TIME())`

Returns: 9

Function: IF

Syntax: IF(condition,statement1,statement2)

Explanation: The IF function evaluates a condition and executes one of two different statements depending on whether the condition is met (true) or not met (false).

Parameter: The following parameters are available

Parameter	Description
condition	A numeric or string expression that evaluates to True or False .
statement1	a single statement; executed if <i>condition</i> is True .
statement2	a single statement; executed if <i>condition</i> is False .

Notes:

- Nested IF statements are supported
- statement2 is OPTIONAL

See also: None

Examples

Enter: IF([Salary1]>99999,SETPROPERTY(vfVisible,"1"),SETPROPERTY(vfVisible,"0"))

Returns: The above IF statement is added to the “CheckSalary” field.
In the above example, CONDITION is set to [Salary1]>99999.
If this condition is TRUE, then the first SETPROPERTY statement is executed, which will makes the “CheckSalary” field visible; otherwise the second SETPROPERTY statement is executed, which will make the “CheckSalary” field invisible.

Enter: IF([Salary2]>99999,SETPROPERTY(vfVisible,"1"))

Returns: In theabove example there is no statement to execute for the FALSE condition.

Enter: IF([Salary1]>99999,GOTOFIELD(“Rich”),IF([Salary1]<100,GOTOFIELD(“Poor”),GOTOFIELD(“EverybodyElse”)))

Returns: The nested IF statement moves the cursor to the appropriate field, depending on the salary.

Function: LEFT

Syntax: LEFT(string,length)

Explanation: Returns a string containing a specified number of characters from the left side of a string.

Parameter: The following parameters are available

Parameter	Description
string	String expression from which the leftmost characters are returned. If <i>string</i> contains Null, Null is returned.
length	Numeric expression indicating how many characters to return. If 0, a zero-length string("") is returned. If greater than or equal to the number of characters in <i>string</i> , the entire string is returned.

Notes: None

See also: RIGHT(), STR(), STRAT(), STRINSTR(), STRLEN(), UPPER(), LOWER(), STREXTRACT()

Example

Where: [Edit1] contains “*Visual eForms Designer is Powerful*”

Enter: LEFT([Edit1],4)

Returns: “Visu”

Function: **LOWER**

Syntax: `LOWER(String)`

Explanation: Returns the lower case representation of the string.

Parameter: The following parameters are available

Parameter	Description
<code>string</code>	any valid string expression. If <i>string</i> contains Null, Null is returned

Notes: None

See also: LEFT(), RIGHT(), STR(), STRAT(), STRINSTR(), STRLEN(), UPPER(), STREXTRACT()

Example

Where: [Edit1] contains “*VISUAL EFORMS ENTERPRISE SERVER*”

Enter: `LOWER([Edit1])`

Returns: “visual eforms enterprise server”

Function: LTRIM

Syntax: LTRIM(String)

Explanation: Removes the leading spaces from a string.

Parameter: The following parameters are available

Parameter	Description
string	any valid string expression. If <i>string</i> contains Null, Null is returned

Notes: LTRIM() returns a string.

See also: RTRIM()

Example

Where: [Edit1] contains “ *Visual eForms Toolbox is superior.* ”

Enter: LTRIM([Edit1])

Returns: “*Visual eForms Toolbox is superior.*”

Function: **MINUTE**

Syntax: `MINUTE (time)`

Explanation: Returns the minute portion of the time from a string containing HH:MM:SS.

Parameter: The following parameters are available

Parameter	Description
<code>time</code>	any expression that can represent a time. If <i>time</i> contains Null, Null is returned

Notes: MINUTE() function returns a numeric value

See also: DATE(), TIME(), DAY(), MONTH(), YEAR(), HOUR(), SEC()

Examples

Where: [Edit1] contains "10:15:22"

Enter: `MINUTE ([Edit1])`

Returns: 15

Enter: If current time is 10:23 then
`MINUTE (TIME ())` returns 23

Enter: "10:25:22"

Returns: 25

Function: **MONTH**

Syntax: MONTH(*date*)

Explanation: Returns the month portion of the date from a string containing MM/DD/YYYY.

Parameter: The following parameters are available

Parameter	Description
<i>date</i>	any expression that can represent a date. If <i>date</i> contains Null, Null is returned

Notes: MONTH() function returns a number.

See also: DATE(), TIME(), DAY(), YEAR(), HOUR(), MINUTE(), SEC()

Example

Where: [Edit1] contains "03/22/2001"

Enter: MONTH([Edit1])

Returns: 3

Enter: If current date is February 28, 2004 then
MONTH (DATE()) returns 2

Enter: MONTH("12/04/2001")

Returns: 12

Function: **NUM**

Syntax: `NUM(string)`

Explanation: Converts an expression into a numeric value.

Parameter: The following parameters are available

Parameter	Description
string	any valid expression

Notes: Use equal number of left and right parentheses.

Returns a numeric value.

NUM rounds to the number of decimal points specified in the decimal points property of the receiving number field of a calculation.

See also: `STR()`

Examples

Where: [Edit1] contains “20”, [Edit2] contains “3”
[Edit3] contains “4”, [Edit5] contains “2”

Enter: `NUM([Edit1])+NUM([Edit2])+NUM([Edit3])`

Returns: 27

NUM converts the strings “20”, “3”, and “4” to numbers and then adds them.

Enter: `[Edit1]+[Edit2]+[Edit3]`

Returns: “2034”

Contents of fields [Edit1], [Edit2] and [Edit3] are concatenated. NUM is not used, and the strings are placed one after another in sequence.

Enter: `NUM([Edit5]+“4”)`

Returns: 24

Content of field [Edit5] is concatenated with string “4” resulting a new string: “24”. NUM is then used to convert the string to a number and the result is number 24.

Enter: `NUM([Edit5]+“sun”)`

Returns: 2

Content of field [Edit5] is concatenated with the string “sun” resulting a new string: “2sun”. NUM returns the number portion of the “2sun” string and 2 is the only number in the string.

Function: PRINTDIALOG

Syntax: PRINTDIALOG()

Explanation: Displays a Print Dialog allowing the user to first select a Printer Destination and then print the currently loaded form based on the Print Dialog settings.

Parameters: None.

Remarks: None.

Return Values: None.

Example:

- Set the OnClick action of a button to PRINTDIALOG()
- Go to Fill Mode
- Click on the button, and the print dialog appears.

Function: **RIGHT**

Syntax: `RIGHT(string, length)`

Explanation: Returns a string containing a specified number of characters from the right side of a string.

Parameter: The following parameters are available

Parameter	Description
<code>string</code>	String expression from which the rightmost characters are returned. If <i>string</i> contains Null, Null is returned
<code>length</code>	Numeric expression indicating how many characters to return. If 0, a zero-length string is returned. If greater than or equal to the number of characters in <i>string</i> , the entire string is returned.

Notes: None

See also: LEFT(), STR(), STRAT(), STRINSTR(), STRLEN(), UPPER(), LOWER(), STREXTRACT()

Example

Where: [Edit1] contains “*Visual eForms Designer can import your forms*”

Enter: `RIGHT([Edit1], 3)`

Returns: “rms”

Function: **ROUND**

Syntax: `ROUND (expression , decimalplaces)`

Explanation: Rounds a number to the number of decimal places.

Parameter: The following parameters are available

Parameter	Description
expression	Required. Numeric expression being rounded
decimalplaces	Optional. Number indicating how many places to the right of the decimal are included in the rounding. If omitted, no rounding takes place.

Notes: Returns a numeric value

See also: None

Examples

Enter: `ROUND (1234.561)`
no rounding, truncates to whole number

Returns: displays "1234"

Enter: `ALERT (STR (ROUND (1,234.561, 2))`
rounds up or down by 2 digits precision factor

Returns: displays "1,234.56"

Function: **RTRIM**

Syntax: `RTRIM(String)`

Explanation: Removes the trailing spaces from the string.

Parameter: The following parameters are available

Parameter	Description
<code>string</code>	any valid string expression. If <i>string</i> contains Null, Null is returned

Notes: RTRIM() function returns a string

See also: LTRIM()

Example

Where: [Edit1] contains “*Visual eMerge merges forms and data*”

Enter: `RTRIM([Edit1])`

Returns: “*Visual eMerge merges forms and data*”

Function Name: **SEC**

Syntax: `SEC(time)`

Explanation: Returns the seconds portion of the time from a string containing HH:MM:SS.

Parameter: The following parameters are available

Parameter	Description
<code>time</code>	any expression that can represent a time. If <i>time</i> contains Null, Null is returned.

Notes: SEC() function returns a numeric value

See also: DATE(), TIME(), DAY(), MONTH(), YEAR(), HOUR(), MINUTE()

Examples

Where: [Edit1] contains "10:15:22"

Current time is: "04:20:33"

Enter: `SEC([Edit1])`

Returns: 22

Enter: If current time is 04:20:33 then

`SEC (TIME())` returns 33

Enter: `SEC("10:15:09")`

Returns: 9

Enter: `if (HOUR(TIME())>9, STR(HOUR(TIME())), "0"+STR(HOUR(TIME())))+if (MINUTE(TIME())>9, STR(MINUTE(TIME())), "0"+STR(MINUTE(TIME())))+if (SEC(TIME())>9, STR(SEC(TIME())), "0"+STR(SEC(TIME())))`

Returns: "042033"

Function: **SETFIELDDATA**

Syntax: SETFIELDDATA(FieldName, FieldData)

Explanation: Sets the value of field FieldName to FieldData.

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field
FieldData	New Data for field FieldName

Remarks: None.

Return Values: None

Example: SETFIELDDATA("Currency1","1234.56")

This example will put "1234.56" into the "Currency1" field.

SETFIELDDATA("Currency1","This is a joke")

This example will try to put "This is a joke" into the "Currency1" field, but fails because "Currency1" is a Numeric field. The value of "Currency1" is set to "0".

Function: **SETFIELDPROPERTY**

Syntax: SETFIELDPROPERTY(FieldName, PropertyID, PropertyVal)

Explanation: Make changes to the properties of a field, one property at a time.

Parameter: The following parameters are available

Parameter	Description
FieldStr	Name of the field on the form
PropertyIDStr	One of the following properties: vfBackColor, vfBottomBorder, vfEnabled, vfLeftBorder, vfLineColor, vfMaxFillChar, vfPageNumber, vfRightBorder, vfRoundedBorder, vfTextColor, vfTopBorder, vfVisible
PropertyVal	A string value. "1" or "0" for ON/OFF, "255,0,255" for setting the RGB color values.

Remarks:

Return Values: None

Example:

```
SETFIELDPROPERTY("SSN",vfVisible,"0")
```

Make the field "SSN" invisible

```
SETFIELDPROPERTY("SSN", vfBackColor,"255,0,0")
```

sets the background color of field "SSN" to **RED**

```
SETFIELDPROPERTY("SSN", vfRoundedBorder,"1")
```

changes the corner borders of field "SSN" to round corners.

Function: **SETPROPERTY**

Syntax: `SETPROPERTY(PropertyId,Property Values)`

Explanation: Changes the property of the field that is calling this script.

Parameter: The following parameters are available

Parameter	Description
PropertyIDStr	One of the following properties: vfBackColor, vfBottomBorder, vfEnabled, vfLeftBorder, vfLineColor, vfMaxFillChar, vfPageNumber, vfRightBorder, vfRoundedBorder, vfTextColor, vfTopBorder, vfVisible
PropertyVal	A string value. "1" or "0" for ON/OFF, "255,0,255" for setting the RGB color values.

Examples

Where: [Salary1] contains \$125,000
[A] contains 1
[B] contains 5

Enter: `IF([A]+[B]>4,SETPROPERTY(vfBackColor,"255,0,0"))`

Returns: If adding the contents of A and B returns a number greater than 4, then the background color property of the field is changed to RED

Enter: `IF([Salary1]>99999,SETPROPERTY(vfTextColor,"255,0,0"))`

Returns: If the content of field "Salary1" is greater than 99999, then the font color of the field is changed to RED

Enter: `IF([Salary1]>99999,SETPROPERTY(vfTextColor,"255,0,0"),SETPROPERTY(vfBackColor,"0,255,0"))`

Returns: If the content of field "Salary1" is greater than 99999, then change the font color of the field to **RED**; otherwise change the bgcolor to **GREEN**.

Function: **STR**

Syntax: `STR(Number, decimal places)`

Explanation: Returns a string representation of a number.

Parameter: The following parameters are available

Parameter	Description
Number	any valid expression

Notes: Accepts an optional second parameter for the number of digits after a decimal point to be converted to string.

See also: `NUM()`

Examples

Enter: `STR(1+2)`

Returns: `"3"`

Enter: `STR(54321)`

Returns: `"54321"`

Enter: `STR(12345.678, 2)`

Returns: `"12345.67"`

Function: **STRAT**

Syntax: `STRAT(string, startposition)`

Explanation: Returns a string beginning at the specified position.

Parameter: The following parameters are available

Parameter	Description
<code>string</code>	String expression being searched
<code>startposition</code>	Numeric expression that sets the starting position for the search

Notes: "Startposition" is offsetted at zero meaning that the first character of "string" is at position zero, the next character is at position 1, etc.

See also: LEFT(), RIGHT(), STR(), STREXTRACT(), STRINSTR(), STRLEN(), UPPER(), LOWER()

Example

Where: [Edit1] contains *"Visual eForms Enterprise Server"*

Enter: `STRAT([Edit1], 3)`

Returns: *"ual eForms Enterprise Server"*

Note that "V" is in position zero, "i" is position 1, "s" is position 2, and "u" is position 3. In the above example STRAT() returns a string starting at "u", which is in position 3.

Enter: `STRAT("January 2000", 6)`

Returns: *"y 2000"*

The "J" is in position 0, the "a" is position 1, the "n" is position 2, etc.

Function: **STREXTRACT**

Syntax: `STREXTRACT(string, startposition, length)`

Explanation: Returns a string beginning at the specified position and for the specified length.

Parameter: The following parameters are available

Parameter	Description
string	String expression being searched
startposition	Numeric expression that sets the starting position for the search
length	Numeric expression that sets the number of characters to return

Notes: "startposition" is offsetted at zero meaning that the first character of "string" is at position zero, the next character is at position 1, etc.

See also: LEFT(), RIGHT(), STR(), STRAT(), STRINSTR(), STRLEN(), UPPER(), LOWER()

Examples

Where: [Edit1] contains "0123456789"

Enter: `STREXTRACT([Edit1], 3, 4)`

Returns: "3456"

In the above example startposition is set to 3, which points to the 4th character from left of the string (i.e. "3").

Enter: `STREXTRACT("exceptional", 2, 3)`

Returns: "cep"

In the above example startposition is set to 2, which points to the 3rd character from left of the string (i.e. "c").

Function: **STRINSTR**

Syntax: `STRINSTR(string1,string2)`

Explanation: Returns the position of the first occurrence of one string within another.

Parameter: The following parameters are available

Parameter	Description
<code>string1</code>	String expression being searched
<code>string2</code>	String expression searched for

Notes: None

See also: `LEFT()`, `RIGHT()`, `STR()`, `STRAT()`, `STRLEN()`, `UPPER()`, `LOWER()`, `STREXTRACT()`

Examples

Enter: `STRINSTR("forgotten","g")`

Returns: "gotten"

Where: [Edit1] contains "Enterprise Server"

Enter: `STRINSTR ([Edit1],"s")`

Returns: "se Server"

Function: **STRLEN**

Syntax: `STRLEN(String)`

Explanation: Returns the number of characters in a string.

Parameter: The following parameters are available

Parameter	Description
String	Any valid string expression. If <i>string</i> contains Null, 0 is returned

Notes: STRLEN() function returns a numeric value

See also: LEFT(), RIGHT(), STR(), STRAT(), STRINSTR(), UPPER(), LOWER(), STREXTRACT()

Examples

Where: [Edit1] contains “*Visual eForms*”
[Edit2] contains “*Toolbox*”

Enter: `STRLEN([Edit1])`

Returns: 13

Enter: `STRLEN("This is a test")`

Returns: 14

Enter: `STRLEN([Edit1]) + STRLEN([Edit2])`

Returns: 20

Function: **SUM**

Syntax: SUM (ColumnName)

Explanation: Returns the sum of *all* cells in a table column, excluding the header, if applicable. Cells should be Number or Currency.

Parameter: The following parameters are available

Parameter	Description
ColumnName	Name of the column in the Table object

- Notes:**
- SUM() function applies only to Table objects.
 - If ColumnName is the name of an object other than a table column, SUM function will return zero.
 - Cells within a table column are indexed from 1 to n (where n is the number of rows in the table).
 - An alternative to SUM() is to add individual cells of a column.
For Example: [Cost:1]+[Cost:2]+[Cost:4]
 - Calculations that involve table cells can use a wild character (i.e. *). Instead of separate calculations such as TOTAL:1= PRICE:1 + TAX:1 for row #1 and TOTAL:2= PRICE:2 + TAX:2 for row #2, use the same syntax on each row: TOTAL:* = PRICE:* + TAX:*. Filler will internally replace * with the proper row number.

Enter: SUM ([Cost1])

Returns: The total of every cell in the [Cost1] column.

Plants	Quantity	Cost
Hawaiian Palms	2.00	\$9.00
Banana Palms	1.00	\$11.00
Orchids	3.00	\$13.00
Ferns	2.00	\$12.00
Hitiscus	0.00	\$0.00
Other	4.00	\$9.00
		\$54.00

Function: **SUMDATE**

Syntax: `SUMDATE(DateExpr, NumExpr)`

Explanation: Calculates a date that is a specified number of days between or after another date.

Notes:

- `SUMDATE()` function returns a string.
- `DateExpr` is a string
- `NumExpr` is a numeric value
- If `NumExpr` is a negative value, the calculated date is before `DateExpr`.
If `NumExpr` is a positive value, the calculated date is after `DateExpr`.

See also: `DATE()`, `DAY()`, `MONTH()`, `YEAR()`, `HOUR()`, `MINUTE()`, `SEC()`, `DIFFDATE()`, `DIFFTIME()`, `SUMTIME()`

Example

Where: Current date is "02/14/2004"

Enter: `SUMDATE(DATE(),4)`

Returns: "02/18/2004"

Enter: `SUMDATE("03/30/2004",5)`

Returns: "04/04/2004"

Enter: `SUMDATE(DATE(),-4)`

Returns: "02/10/2004"

Function: **SUMTIME**

Syntax: `SUMTIME(TimeExpr , NumExpr)`

Explanation: Calculates a time that is a specified number of hours between or after another time.

Notes:

- TimeExpr is a string
- NumExpr is a numeric value
- SUMTIME() function returns a string.
- If NumExpr is a negative value, the calculated time is before TimeExpr.
If NumExpr is a positive value, the calculated time is after TimeExpr.

See also: DATE(), DAY(), MONTH(), YEAR(), HOUR(), MINUTE(), SEC(),
DIFFTIME(), DIFFDATE(), SUMDATE()

Example

Where: Current time is "04:20:08"

Enter: `SUMTIME(TIME(),5)`

Returns: "09:20:08"

Enter: `SUMTIME("11:20:53",3)`

Returns: "14:20:53"

Enter: `SUMTIME(TIME(),-2)`

Returns: "02:20:08"

Function: **TIME**

Syntax: `TIME ()`

Explanation: Returns the current system time in HH:MM:SS: format.

Notes: `TIME()` function returns a string.

`TIME()` function is an automatic script. The value of the field will automatically be set to current system time.

See also: `DATE()`, `DAY()`, `MONTH()`, `YEAR()`, `HOUR()`, `MINUTE()`, `SEC()`

Example

Where: Current time is "04:20:08"

Enter: `TIME()`

Returns: "04:20:08"

Enter: `IF (HOUR(TIME())>9, STR(HOUR(TIME())), "0"+STR(HOUR(TIME())))+IF (MINUTE(TIME())>9, STR(MINUTE(TIME())), "0"+STR(MINUTE(TIME())))+IF (SEC(TIME())>9, STR(SEC(TIME())), "0"+STR(SEC(TIME())))`

Returns: "042008"

Function: **UPPER**

Syntax: `UPPER (String)`

Explanation: Returns a string that has been converted to uppercase.

Parameter: The following parameters are available

Parameter	Description
String	any valid string expression. If <i>string</i> contains Null, Null is returned.

Notes: None

See also: LEFT(), RIGHT(), STR(), STRAT(), STRINSTR(), STRLEN(), LOWER(), STREXTRACT()

Example

Where: [Edit1] contains “*Visual eForms is Great*”

Enter: `UPPER ([Edit1])`

Returns: “VISUAL EFORMS IS GREAT”

Function: YEAR

Syntax: YEAR(*String*)

Explanation: Returns a whole number representing the year.

Parameter: The following parameters are available

Parameter	Description
String	any expression that can represent a date in MM/DD/YYYY format. If <i>date</i> contains Null, Null is returned.

Notes: YEAR() function returns a numeric value

See also: DATE(), TIME(), DAY(), MONTH(), HOUR(), MINUTE(), SEC()

Examples

Where: [Edit1] contains “03/22/2005”
The current year is 2005

Enter: YEAR([Edit1])

Returns: 2005

Enter: YEAR(“03/22/2005”)

Returns: 2005

Enter: YEAR (DATE())

Returns 2005

Databases

Database Connectivity

A database is a collection of data records that can be used and updated by different forms and different applications. Visual eForms forms can connect to several different types of databases in one of two ways:



ActiveX is a
component
based
standard

created by Microsoft.

Please refer to sample applications to learn more about database connectivity.

- Coding

- MmaADOi - Multimedia Abacus Database ActiveX

Coding

By coding around the Filler ActiveX control methods within a given programming environment, you can connect to any data source. Using third-party database connectivity software makes this process even easier. Coding is best suited for users with a good programming background and previous database programming experience.

MmaADOi

MmaADOi is built on top of the Microsoft ActiveX Data Objects (ADO). The ActiveX Control, when used in conjunction with the database relations function of the Visual eForms Designer, makes it easy to connect forms to a variety of databases.

The ADOi currently supports the following database types:

- MS Access
- ODBC
- dBASE III
- dBASE IV
- dBASE 5
- Paradox 3.x
- Paradox 4.x
- Paradox 5.x
- FoxPro 2.0
- FoxPro 2.5
- FoxPro 2.6
- Excel 3.0
- Excel 4.0
- Excel 5.0
- Excel 95
- Excel 97
- Text

Note: *MmaADOi Properties and Methods are discussed in “Database ActiveX” on page 335*

Choosing a Database Format

As mentioned previously, a database is a collection of data records that can be used and updated by different forms and different applications. Visual eForms forms can connect to several different types of databases.

>When choosing a database, consider these factors:

- Data storage standards (to interface with other widely used applications).
- Formats that others are using to fill the same form.
- Data storage capacity. Some formats require more disk space than others.
- Applications that you may need to perform independently from the form.

Database Relations

Database relations is the mechanism of assigning fields on the forms to fields in databases. Before creating database relations, you must first save the form.

>To save a form

Quick key:
[CTRL]+[S]

On the **File** menu, click **Save**.

-or-

Click the **Save** icon on the toolbar.

Database Relations Screen

The **Database Relations Screen** provides a graphic representation of a form's database structure in which you can drag and drop fields to create links between form fields and data sources.

- Data sources are tables in databases that contain information. For example, in a Microsoft Access database, you may have several tables such as a name table, an address table, and an occupation table.

From the **Database Relations** screen you can:

- assign or reassign form fields to data sources.
- define new data sources.
- assign existing data sources to your form.
- define relationships between multiple data sources.

Data Sources can be created using CreateDatabase() method of Database ActiveX. See "CreateDatabase" on page 339

>To open the Database Relations screen

Click the **Database Relations** icon on the **Standard** toolbar.

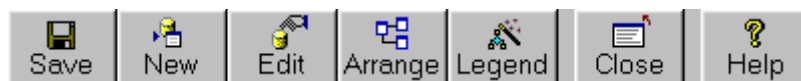
-or-

Quick key: [ALT] + [D]

On the **File** menu, click **Database Relations**.

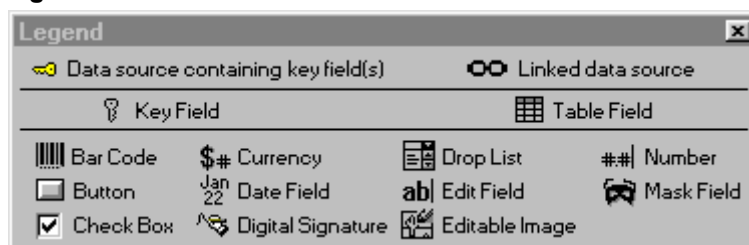
The **Database Relations** screen appears.

Toolbar



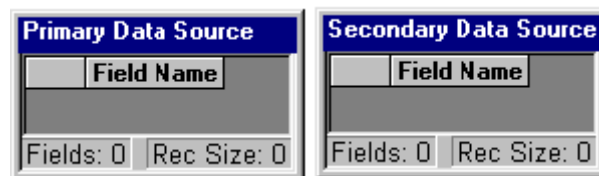
The toolbar allows you to **Save**, **Edit**, **Arrange**, and **Close** the data source relations that you create. It also allows you to receive **Help**, create **New** data sources, and display the **Legend**.

Legend



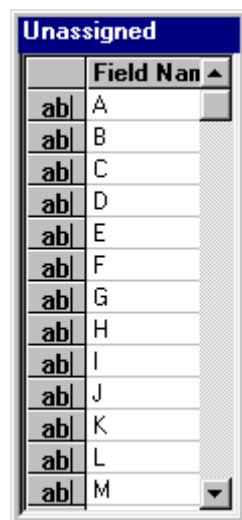
The **Legend** explains the symbols used in this screen.

Primary and Secondary Data Source Windows



The **Primary Data Source** window is an individual data source window, as is the **Secondary Data Source** window.

Unassigned Column



The unassigned column shows which form fields are not assigned to data sources.

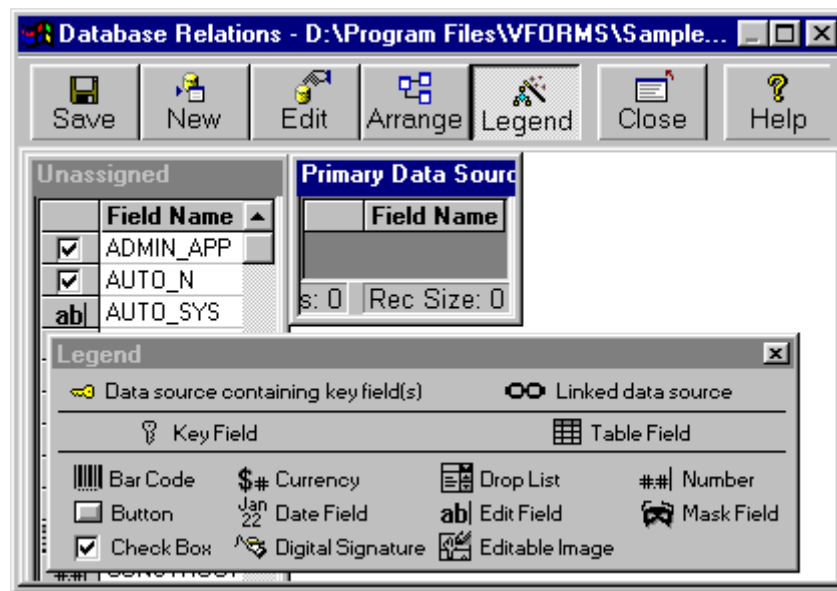
Primary and Secondary Data Sources

You can create primary and optional secondary data sources and assign fillable form objects to those data sources.

>To create a data source

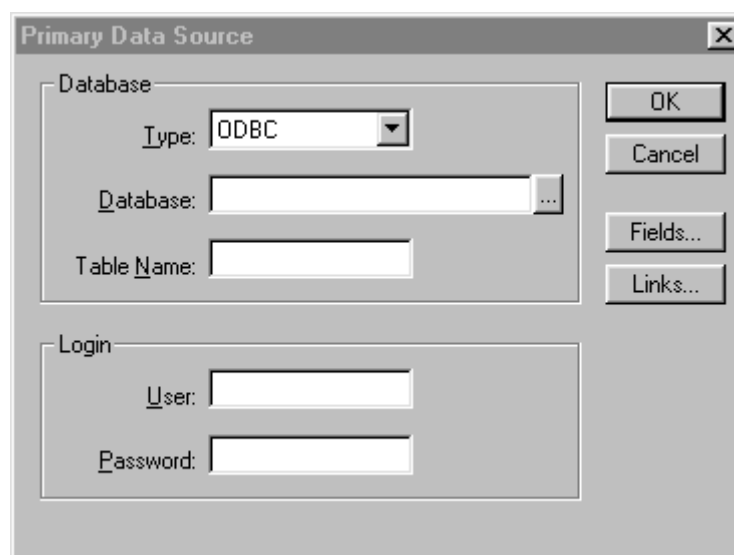
1. Open the **Database Relations** screen.

The **Database Relations** screen appears.



2. Double-click on the **Primary Data Source** box.

The **Primary Data Source** dialog box appears.



3. In the **Database** section, click the **Type** drop-down menu to select the type of database.

You can choose from databases that were listed in “MmaADOi” on page 148.

4. Click the browse icon next to the **Database** field.

A dialog box appears allowing you to browse through data sources of the type that you have just selected.

5. In the dialog box, double-click on the specific data source that you would like to use.

The data source file appears in the **Database** field.

The screenshot shows a Windows-style dialog box titled "Secondary Data Source (2)". It contains two main sections: "Database" and "Login". In the "Database" section, the "Type" is set to "MS Access" in a dropdown menu. The "Database" field shows the path "D:\Program Files\WFORMS\Dat" with a browse button (three dots) to its right. The "Table Name" field is empty. In the "Login" section, the "User" and "Password" fields are empty. At the bottom left, there is a checkbox labeled "Cascade Delete" which is currently unchecked. On the right side of the dialog, there are four buttons: "OK", "Cancel", "Fields...", and "Links...".

6. In the **Table Name** field, enter a table name from the data source that is referenced (e.g., Name, Address, or State).
7. If necessary, enter your **User** name and **Password** in the **Login** section.
8. Click the **OK** button.

A secondary data source is optional. If your form requires you to use more than one data source, you can create one.

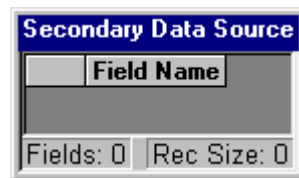
>To create a secondary data source

1. Click the **New** icon.

A **Secondary Data Source** dialog box appears.

2. Repeat the same steps to select a secondary data source as you did to select a primary data source.

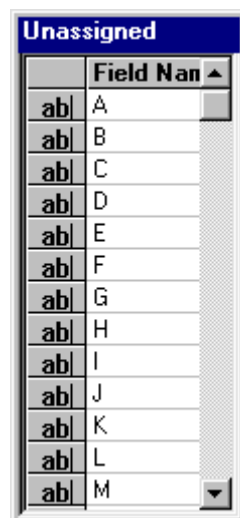
The **Secondary Data Source** window appears.



Assigning Database Relations

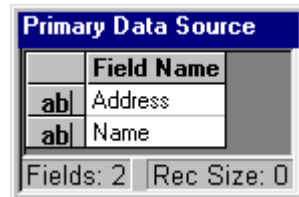
After designing your form fields, use the **Database Relations** screen to assign the appropriate databases to your form.

The **Unassigned** window contains all of the fields in your form that have no assigned relationships to data sources.

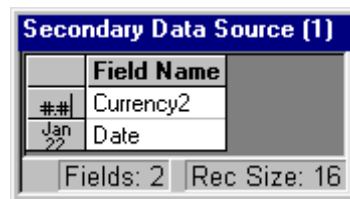


>To assign fields to the primary data source

1. Click a field in the **Unassigned** column.
2. Hold down the mouse button and drag the field to the **Primary Data Source** window.
3. Repeat this process until all of the desired fields are contained in the **Primary Data Source** window.

**>To assign fields to the Secondary Data Source**

1. Click a field in the **Unassigned** column.
2. Hold down the mouse button and drag the field to the **Secondary Data Source** window.

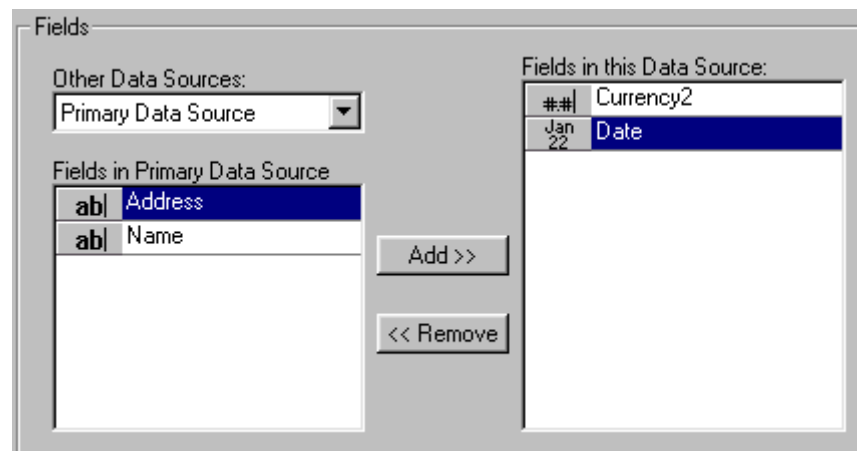


3. Repeat this process until all of the desired fields are contained in the **Secondary Data Source** window.

You can also assign fields in the **Data Source** dialog box.

>To assign fields in the Data Source dialog box

1. Right-click the **Primary** or **Secondary Data Source** window.
2. Select **Properties** from the menu.
3. When the dialog box appears, click the **Fields** icon.
4. In the fields section, select the data source from the **Other Data Sources** drop-down menu.

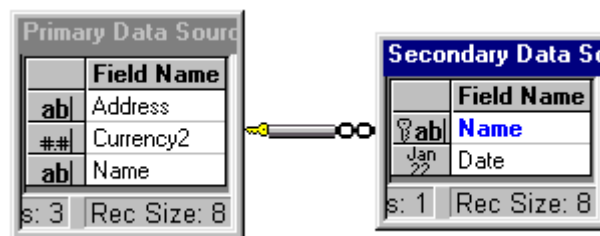


5. Select a field in either **Data Source** field and click **Add** or **Remove**.
6. Click the **OK** button when you have finished defining the Link Fields.

>To link the Primary and Secondary Data source boxes

1. Click the field that you want to define as the link key in the **Primary Data Source** window.
2. Hold down the [CONTROL] key while dragging the selected field to the **Secondary Data Source** window.

A chain, bar, and key denote that the fields are linked. The highlighted field name with the key icon next to it indicates which field in the table is used to link the two data sources.

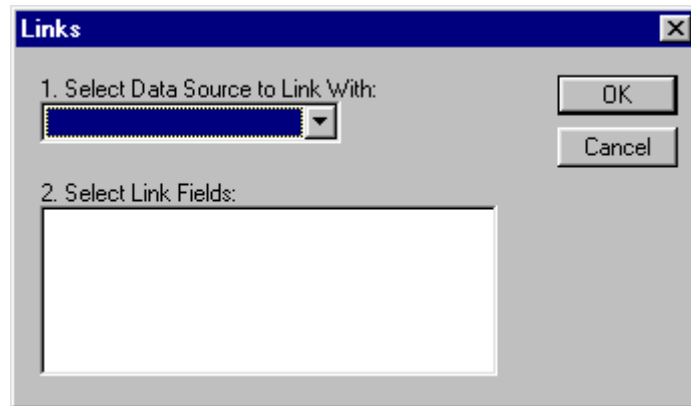


Repeat this process for any further links. You can link a primary to a secondary data source, or link two secondary data sources.

You can also define or change links in the **Data Source** dialog box.

>**To define links in the Data Source dialog box**

1. Right-click the **Primary** or **Secondary Data Source** window.
2. Select **Properties** from the menu.
3. When the dialog box appears, click **Links**.



4. Select the data source from the **Select Data Source to Link With** drop-down menu.
5. Select a field within **Select Link Fields**.
6. Click the **OK** button when you have finished selecting the links.

Using Database Relations

>To see how the Database Relations function works with forms

1. Open a container application.

A container application is a programming environment capable of containing ActiveX controls. These may include any of the following:

- Visual Basic
 - Visual C++
 - Delphi
 - HTML
 - Lotus Notes
 - FrontPage
2. Drop the Filler ActiveX control and the Database ActiveX control into your container.
 3. Add an **Open** icon.
 4. Add the connectivity code.
 5. Click the **Open** icon to view the form that you have just connected.

Example of code in Visual Basic



For information on syntax, See "Database ActiveX" on page 337.

```
dim rv  
rv = MmaDbase1.Connect(App.Path & "\test.far",  
MmaFill1, " ")
```

Advanced Programming

Visual eForms is a component-based electronic forms engine built around the Microsoft ActiveX technology and Microsoft Foundation Classes (MFC). The component approach allows for ease of programming and integration with popular programming environments such as Visual Basic, Visual C++, Delphi, PowerBuilder, and Lotus Notes.

In general, Visual eForms Filler ActiveX control can be integrated into any 32-bit Windows application, which can act as an ActiveX Control Container.

ActiveX control can also be integrated into HTML to empower and add additional functionality to it. This very important capability enables you to develop powerful Web-based applications using Cerenade's Filler Control.

Sample codes provided in this chapter are written in Visual Basic. An understanding of Visual Basic or a similar programming language is necessary to understand the methods discussed in this chapter.

In all functions where `FieldName` can be used you can replace the `FieldName` with the TAB ORDER of the field, if known. You would then put a "@" character in front of it. Examples:

- `GotoField ("@12")` which takes you to the field that is on Tab Order 12 . or
- `GotoField ("SSN")` which takes you to the field that is named "SSN"

Note that if "@" is missing, the function will act on a field that is named "12", which can produce different results.

This section contains the lists of all Visual eForms properties and methods built around Filler ActiveX.

Filler Active X

Properties

DefaultPath

Description:

Gets or sets the Default Path associated with the ActiveX control.

Syntax:

```
C++          CString CMmaFill::GetDefaultPath ()  
              void CMmaFill::SetDefaultPath (LPCTSTR DefaultPath)  
  
Visual Basic [form.]MmaFill. DefaultPath
```

Remarks:

By default, objects referring to filenames (e.g., an Image object referring to a .bmp or .wmf file) without a full pathname are resolved relative to the directory of the application.

You may use this property to change the default path to an alternate address (another directory on the network or a URL on the internet).

Data Type:

String

FormName

Description:	Gets or sets the Form Name associated with the currently loaded form.
Syntax:	<div>C++ <code>CString CMmaFill::GetFormName()</code> <code>void CMmaFill::SetFormName(LPCTSTR FormName)</code></div> <div>Visual Basic <code>[form.]MmaFill. FormName</code></div>
Remarks:	You may use this property to identify the name of a form within the context of an application.
Data Type:	String

FormVersion

Description:	Gets or sets the Form Version associated with the currently loaded form.
Syntax:	<div>C++ <code>CString CMmaFill::GetFormVersion()</code> <code>void CMmaFill::SetFormVersion(LPCTSTR FormVersion)</code></div> <div>Visual Basic <code>[form.]MmaFill. FormVersion</code></div>
Remarks:	You may use this property to identify the revision of a form within the context of an application.
Data Type:	String

LastErrorCode

Description: Get or Set the error code of last unsuccessful ActiveX operation.

Syntax: C++ long GetLastErrorCode()
Visual Basic [form.]MmaFill.LastErrorCode

Remarks: None

Data Type: Long

See Also: LastErrorDesc

Error Code	Error Description
1000	an invalid property ID was specified
1001	a passed field name was not found within the current form
1002	the operation requires a fillable field
1003	a file was not found
1004	a file could not be created
1005	an attempt was made to duplicate a field name
1006	no form was loaded
1007	an exception occurred
1008	an invalid date and/or time was specified
1009	an operation could not be performed since the form was not saved
1010	an invalid value was specified
1011	an attempt was made to modify a read only property
1012	a file name was needed but missing
1013	invalid name for a form object
1050	error drawing barcode
1051	barcode library not found or unavailable
1075	error parsing
1076	error with parsing table

Error Code	Error Description
1077	error with field validation
1100	not enough colors to contrast
1101	embedding an image failed
1125	need newer version of software to open the specified file
1150	a form archive is corrupt
1151	error reading from a temp file
1175	error attempting to encrypt data
1176	error attempting to decrypt data
1177	crypto library not found or unavailable
1178	cannot access "MY" cert store
1179	no certificate found to sign with
1180	certificate was not issued by a trusted entity
1181	certificate was self signed
1182	an error occurred while performing an Entrust function
1183	CRL could not be retrieved
1184	certificate is not valid because of a date problem
1185	certificate is revoked
1200	error creating a font
1225	an invalid date was entered

LastErrorDesc

Description: Error description of last unsuccessful operation.

Syntax: C++ CString GetLastErrorDesc()
Visual Basic [form.]MmaFill.LastErrorDesc

Remarks: None.

Data Type: String

See Also: LastErrorCode

vfBackColor

Description: ID for Background Color property of a field.

Syntax: C++ long GetVfBackColor()
 voidSetVfBackColor(long)
Visual Basic [form.]MmaFill.vfBackColor

Remarks: Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Background Color property of an object.

Data Type: Long

Property ID: 8

Example: The following Visual Basic code returns the value of this property.

Msgbox (mmafill.VfBackColor)

returns: 8

To turn the background color of field “PhoneNumber” to RED use this Visual Basic code:

rv = mmafill.SetFieldProperty(“PhoneNumber”,8,”255,0,0”)

- or -

rv =

mmafill.SetFieldProperty(“PhoneNumber”,mmafill.VfBackColor,”255,0,0”)

vfBottomBorder

Description:	ID for Bottom Border property of a field.
Syntax:	<pre>C++ long GetVfBottomBorder () void SetVfBottomBorder (long) Visual Basic [form.]MmaFill.vfBottomBorder</pre>
Remarks:	Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Bottom Border of a field.
Data Type:	Long
Property ID:	13
Example:	<p>The following Visual Basic code returns the value of this property.</p> <pre>Msgbox (mmafill.vfBottomBorder) returns: 13</pre> <p>To turn ON the bottom border of field "PhoneNumber" use this Visual Basic code:</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",13,"1")</pre> <p style="text-align: center;">- or -</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfBottomBorder,"1")</pre>

vfButton

Description: Property ID of a Button field

Syntax: C+ long GetVfButton ()
 void SetVfButton (long)
Visual Basic [form.]MmaFill.vfButton

Remarks: Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about object type.

Data Type: Long

Property ID: 32

Example: In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is a button object then “do something”.

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = CStr(mmafill.vfButton) then
’do something
End If
```

or

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = “32” then
’do something
End If
```

vfCheckBox

Description:	Property ID of CheckBox field
Syntax:	<div>C++<pre>long GetVfCheckBox () void SetVfCheckBox (long)</pre></div> <div>Visual Basic <code>[form.]MmaFill.vfCheckBox</code></div>
Remarks:	Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about Field type.
Data Type:	Long
Property ID:	16
Example:	<p>In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is a checkbox object then “do something”.</p> <pre>Var FieldType as String FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType) If FieldType = CStr(mmafill.vfCheckBox) then ’do something End If or Var FieldType as String FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType) If FieldType = “16” then ’do something End If</pre>

vfDate

Description:	Property ID of Date field.
Syntax:	<div>C++<pre>long GetVfDate () void SetVfDate (long)</pre></div> <div>Visual Basic <code>[form.]MmaFill.vfDate</code></div>
Remarks:	Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about Field type.
Data Type:	Long
Property ID:	2048
Example:	In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is a date object then “do something”.

```
Var FieldType as String  
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)  
If FieldType = CStr (mmafill.VfDate) then  
    'do something  
End If  
  
or  
  
Var FieldType as String  
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)  
If FieldType = “2048” then  
    'do something  
End If
```

vfEditableImage

Description: Property ID of Editable Image field

Syntax: C++ long GetVfEditableImage ()
 void SetVfEditableImage (long)
Visual Basic [form.]MmaFill.vfEditableImage

Remarks: Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about Field type.

Data Type: Long

Property ID: 512

Example: In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is an editable image object then “do something”.

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = CStr (mmafill.vfEditableImage) then
’do something
End If
```

or

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = “512” then
’do something
End If
```

vfEnabled

Description:	ID for Overwrite Property of a field.
Syntax:	<pre>C++ long GetVfEnabled () void SetVfEnabled (long) Visual Basic [form.]MmaFill.vfEnabled</pre>
Remarks:	Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Overwrite Property of a field.
Data Type:	Long
Property ID:	28
Example:	<p>The following Visual Basic code checks if the field “Cancel” is enabled.</p> <pre>Var FieldState as String FieldState = mmafill.GetFieldProperty(“Cancel”, mmafill.vfEnabled) If FieldState = “28” then ‘field is enabled - i.e. user can type into the field End If</pre>

vfFillableText

Description:	Property ID of Fillable field
Syntax:	C++ long GetVfFillableText () void SetVfFillableText (long) Visual Basic [form.]MmaFill.vfFillableText
Remarks:	Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about Field type.
Data Type:	Long
Property ID:	8
Example:	In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is an edit object then “do something”.

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = CStr (mmafill.vfFillableText) then
'do something
End If

or

Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = “8” then
'do something
End If
```

vfLeftBorder

Description:	ID for Left Border property of a field.
Syntax:	<pre>C++ long GetVfLeftBorder () void SetVfLeftBorder (long) Visual Basic [form.]MmaFill.vfLeftBorder</pre>
Remarks:	Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Left Border of a field.
Data Type:	Long
Property ID:	10
Example:	<p>The following Visual Basic code returns the value of this property.</p> <pre>Msgbox (mmafill.vfLeftBorder) returns: 10</pre> <p>To turn ON the left border of field "PhoneNumber" use this Visual Basic code:</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",10,"1")</pre> <p style="text-align: center;">- or -</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfLeftBorder,"1")</pre>

vfLineColor

Description: ID for Line Color property of a field.

Syntax: C++ long GetVfLineColor ()
 void SetVfLineColor (long)
Visual Basic [form.]MmaFill.vfLineColor

Remarks: Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Line Color of a field.

Data Type: Long

Property ID: 9

Example: The following Visual Basic code returns the value of this property.

Msgbox (mmafill.vfLineColor)

returns: 9

To turn the border color of field "PhoneNumber" to GREEN use this Visual Basic code:

rv = mmafill.SetFieldProperty("PhoneNumber",9,"0,255,0")

- or -

rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfLineColor,"0,255,0")

vfMaxFillChars

Description:	ID for MaxFillChars property of a field.
Syntax:	<div>C++<pre>long GetVfMaxFillChars () void SetVfMaxFillChars (long)</pre></div> <div>Visual Basic <pre>[form.]MmaFill.vfMaxFillChars</pre></div>
Remarks:	<p>Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the MaxFillChars of a field.</p> <p>Set MaxFillChars to Zero for an unlimited number of characters.</p>
Data Type:	Long
Property ID:	27
Example:	<p>The following Visual Basic code returns the value of this property.</p> <pre>Msgbox (mmafill.VfMaxFillChars)</pre> <p>returns: 27</p> <p>Set the maximum number of characters one can type into field "PhoneNumber" to 80:</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",27,"80")</pre> <p style="text-align: center;">- or -</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.VfMaxFillChars ,"80")</pre>

vfNumber

Description: Property ID of Number field

Syntax: C++ long GetVfNumber ()
 void SetVfNumber (long)
Visual Basic [form.]MmaFill.vfNumber

Remarks: Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about Field type.

Data Type: Long

Property ID: 256

Example: In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is a number object then “do something”.

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = CStr (mmafill.vfNumber) then
’do something
End If
```

or

```
Var FieldType as String
FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType)
If FieldType = “256” then
’do something
End If
```


vfPageNumber

Description:	Property ID of Page Number.
Syntax:	<pre>C++ long GetVfPageNumber () void SetVfPageNumber (long) Visual Basic [form.]MmaFill.vfPageNumber</pre>
Remarks:	Must be used with GetFieldProperty() to Get the page number for a field.
Data Type:	Long
Property ID:	40
Example:	<p>The following Visual Basic code returns the value of this property.</p> <pre>Msgbox (mmafill.VfPageNumber) returns: 40</pre> <p>Which page of the form is the field "PhoneNumber" located on?</p> <pre>rv = mmafill.GetFieldProperty("PhoneNumber",40) - or - rv = mmafill.GetFieldProperty("PhoneNumber",mmafill.VfPageNumber)</pre>

vfRightBorder

Description: ID for Right Border property of a field.

Syntax: C++ long GetVfRightBorder ()
 void SetVfRightBorder (long)
Visual Basic [form.]MmaFill.vfRightBorder

Remarks: Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Right Border of a field.

Data Type: Long

Property ID: 12

Example: The following Visual Basic code returns the value of this property.
Msgbox (mmafill.vfRightBorder)
returns: 12

To turn ON the right border of field “PhoneNumber” use this Visual Basic code:

```
rv = mmafill.SetFieldProperty(“PhoneNumber”,12,”1”)
```

- or -

```
rv = mmafill.SetFieldProperty(“PhoneNumber”,mmafill.vfRightBorder,”1”)
```

vfRoundedBorder

Description: ID for Rounded Border property of a field.

Syntax: C++ long GetVfRoundedBorder ()
 void SetVfRoundedBorder (long)
Visual Basic [form.]MmaFill.vfRoundedBorder

Remarks: Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Rounded Border property of a field.

Data Type: Long

Property ID: 14

Example: The following Visual Basic code returns the value of this property.
Msgbox (mmafill.VfRoundedBorder)
returns: 14

To set all orders of field “PhoneNumber” rounded use this Visual Basic code:

```
rv = mmafill.SetFieldProperty(“PhoneNumber”,14,”1”)
```

- or -

```
rv = mmafill.SetFieldProperty(“PhoneNumber”,mmafill.VfRoundedBorder,”1”)
```

vfSignature

Description:	Property ID for Signature field
Syntax:	<div>C++<pre>long GetVfSignature () void SetVfSignature (long)</pre></div> <div>Visual Basic <code>[form.]MmaFill.vfSignature</code></div>
Remarks:	Must be used with GetFieldProperty() to qualify the return value of GetFieldProperty() when inquiring about Field type.
Data Type:	Long
Property ID:	262144
Example:	<p>In the following Visual Basic example, the type of field “Cancel” is examined. If “Cancel” is a signature object then “do something”.</p> <pre>Var FieldType as String FieldType = mmafill.GetFieldProperty(“Cancel”, mmafill.vfType) If FieldType = CStr (mmafill.vfSignature) then ’do something End If</pre>

vfTextColor

Description:	ID for Text Color property of a field.
Syntax:	<div>C++<pre>long GetVfTextColor () void SetVfTextColor (long)</pre></div> <div>Visual Basic <code>[form.]MmaFill.vfTextColor</code></div>
Remarks:	Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Text Color of a field.
Data Type:	Long
Property ID:	26
Example:	<p>The following Visual Basic code returns the value of this property.</p> <pre>Msgbox (mmafill.vfTextColor) returns: 26</pre> <p>To turn the text color of field "PhoneNumber" to BLUE use this Visual Basic code:</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",26,"0,0,255") - or - rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfTextColor,"0,0,255")</pre>

vfTopBorder

Description:	ID for Top Border property of a field.
Syntax:	<pre>C++ long GetVfTopBorder () void SetVfTopBorder (long) Visual Basic [form.]MmaFill.vfTopBorder</pre>
Remarks:	Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Top Border of a field.
Data Type:	Long
Property ID:	11
Example:	<p>The following Visual Basic code returns the value of this property.</p> <pre>Msgbox (mmafill.vfTopBorder) returns: 11</pre> <p>To turn ON the top border of field "PhoneNumber" use this Visual Basic code:</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",11,"1")</pre> <p style="text-align: center;">- or -</p> <pre>rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfTopBorder,"1")</pre>

vfType

Description: TYPE of a field.

Syntax: C++ long GetVfType ()
 void SetVfType (long)
Visual Basic [form.]MmaFill.vfType

Remarks: When used with GetFieldProperty(), vfType returns the type of a field.

Data Type: Long

Return Values:

vfBarCode	BarCode Field
vfButton	Button Field
vfCheckBox	CheckBox Field
vfDate	Date Field
vfDropList	DropList Field
vfEditableImage	EditableImage Field
vfFillableText	FillableText Field
vfMask	Mask Field
vfNumber	Number or Fixed Field
vfSignature	Signature Field

Property ID: 1

Example: The following Visual Basic code tells us what type of field “Phone Number” is.

```
rv = mmafill.GetFieldProperty(“PhoneNumber”,1)
```

- or -

```
rv = mmafill.GetFieldProperty(“PhoneNumber”,mmafill.vfType)
```

vfVisible

Description:	ID for Visible property of a field.
Syntax:	C++ long GetVfVisible () void SetVfVisible (long) Visual Basic [form.]MmaFill.vfVisible
Remarks:	Must be used with GetFieldProperty() or SetFieldProperty() to Get or Set the Visible property of a field.
Data Type:	Long
Property ID:	76
Example:	The following Visual Basic code returns the value of this property. Msgbox (mmafill.vfVisible) returns: 76

To make field "PhoneNumber" non-visible use this Visual Basic code:

```
rv = mmafill.SetFieldProperty("PhoneNumber",76,"0")
```

- or -

```
rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfVisible,"0")
```

To make field "PhoneNumber" visible use this Visual Basic code:

```
rv = mmafill.SetFieldProperty("PhoneNumber",76,"1")
```

- or -

```
rv = mmafill.SetFieldProperty("PhoneNumber",mmafill.vfVisible,"1")
```


ZoomFactor

Description: Gets or sets the Zoom value for the currently loaded form.

Syntax:

C++ short CMmaFill::GetZoomFactor()
 void CMmaFill::SetZoomFactor(LPCTSTR ZoomFactor)

Visual Basic [form.]MmaFill.ZoomFactor

Remarks: The following settings are available for ZoomFactor:

Setting	Constant Description
1	Whole Page - View the entire page in the window.
2	Page Width - View the entire page width in the window
3 to 400	Set the Zoom Factor to the specified percentage (3% to 400%)

Example: The following Visual Basic code sets the Zoom level for the form to 100%.

```
mmafill.ZoomFactor = 100
```

Methods

AbandonChanges

Description:	Resets the internal Modified-Flag of the currently loaded form to False.
Syntax:	<div>C++ BOOL CMmaFill::AbandonChanges();</div> <div>Visual Basic [form.]MmaFill.AbandonChanges()</div>
Parameters:	None
Remarks:	Use this method in conjunction with IsFormChanged() method to notify the user regarding unsaved changes when he/she is attempting to load a new form or is exiting the application.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>

AboutBox

Description: Displays box containing Visual eForms Copyright details.

Syntax: C++ void AboutBox()
Visual Basic [form.]MmaFill.AboutBox()

Parameters: None

Remarks: None

Return Values: None

AppendField

Description: Appends string AppendString to the contents of field FieldName.

Syntax:

```
C++          BOOL CMmaFill::AppendField(LPCTSTR FieldName, LPCTSTR
              AppendString)

Visual Basic  [form.]MmaFill.AppendField(Byval FieldName As String, Byval
              AppendString As String)
```

Parameters: The following parameters are available:

Parameter	Description
FieldName	Name of the Form Field
AppendString	String to be appended to contents of specified Form Field

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Change the value of LAST_NAME field by appending
"ABCDEF" to it.

```
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.AppendField("LAST_NAME", "ABCDEF")
```

AutoReduceFonts

Description: Changes the automatic Font Reduction settings of the Filler control.

Syntax:

C++ `BOOL CMmaFill::AutoReduceFonts(short Points, BOOL Repaint)`

Visual Basic `[form.]MmaFill.AutoReduceFonts(Byval Points As Integer, Byval Repaint As Boolean)`

Parameters: The following parameters are available:

Parameter	Description
Points	# of point sizes to Auto-Reduce the fill-fonts.
Repaint	set to True to effect a repaint

Remarks: Once this method is called, the changes remain in effect for subsequent forms loaded into the filler until the next call to this method.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

```
'Set AutoReduceFont property of the form to 3
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.AutoReduceFonts(3, True)
```

ClearData

Description:	Clears the contents of all fields on the currently loaded form.
Syntax:	<div>C++ BOOL CMmaFill::ClearData() Visual Basic [form.]MmaFill.ClearData()</div>
Parameters:	None.
Remarks:	Before calling this method, it is wise to check for any unsaved changes to the current form and notifying the user of the application, if necessary.
Return Values:	<div>True - Call to this method was Successful False - An Error was encountered</div>
Example:	<div>'First: Fill form fields with test data 'Second: Confirm clearing 'Third: Clear data from all fields on the form Dim rv rv = mmafill.OpenFormDialog() rv = mmafill.FillTestData() rv = MsgBox ("Clearing Data." + Chr(10) + "Are You Sure?", vbYesNo) If rv = vbYes Then mmafill.ClearData() End If</div>

CloseForm

Description:	Closes the currently loaded form and terminates work with it.
Syntax:	<div>C++ BOOL CMmaFill::CloseForm() Visual Basic [form.]MmaFill.CloseForm()</div>
Parameters:	None.
Remarks:	Before calling this method, it is wise to check for any unsaved changes to the current form and notifying the user of the application, if necessary.
Return Values:	<div>True - Call to this method was Successful False - An Error was encountered</div>
Example:	<div>'Close the form Dim rv rv = MmaFill1.CloseForm()</div>

Copy

Description: Copies currently selected text onto the clipboard.

Syntax: C++ `BOOL CMmaFill::Copy()`
Visual Basic `[form.]MmaFill.Copy()`

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Copy the first 5 characters of LAST_NAME field and Paste it into
'LAST_NAME_2 field
'Use DisableRedraw to eliminate flickering, if any
'Enable Redraw after the procedure
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.DisableRedraw(True)
rv = MmaFill1.GotoField("LAST_NAME")
rv = MmaFill1.SetCursorPosition(0, 5)
rv = MmaFill1.Copy()
rv = MmaFill1.GotoField("LAST_NAME_2")
rv = MmaFill1.Paste()
rv = MmaFill1.DisableRedraw(False)

CreateNote

Description: Creates a Sticky Note or TypeAnywhere object on the loaded form.

Syntax:

C++ void CMmaFill::CreateNote(BOOL TypeAnywhere, LPCSTR Options)

Visual Basic [form.]MmaFill.CreateNote(byval bTypeAnywhere As Boolean, byval Options As String)

Parameters: The following parameters are available:

Parameter	Description																																								
bTypeAnywhere	TRUE: Add a TypeAnywhere field FALSE: Add a Sticky Note field																																								
Options	a set of parameters in the following format: option=value;option=value;....;option=value																																								
	<table> <tr> <th><u>option</u></th><th><u>value</u></th></tr> <tr> <td>Opaque</td><td>True or False</td></tr> <tr> <td>Callout</td><td>True or False</td></tr> <tr> <td>Printable</td><td>True or False</td></tr> <tr> <td>Font</td><td>Courier New</td></tr> <tr> <td></td><td>Arial</td></tr> <tr> <td></td><td>etc.</td></tr> <tr> <td>FontFamily</td><td>Swiss</td></tr> <tr> <td></td><td>Roman</td></tr> <tr> <td></td><td>etc.</td></tr> <tr> <td>FontSize</td><td>in units eg. 9.0</td></tr> <tr> <td>FontBold</td><td>1 or True</td></tr> <tr> <td>FontItalic</td><td>1 or True</td></tr> <tr> <td>FontUnderline</td><td>1 or True</td></tr> <tr> <td>Spacing</td><td>0=Single Point size</td></tr> <tr> <td>Border</td><td>1 or True</td></tr> <tr> <td>TextColor</td><td>RGB value eg. (255,0,255)</td></tr> <tr> <td>BackColor</td><td>RGB value</td></tr> <tr> <td>CalloutColor</td><td>RGB value</td></tr> <tr> <td>CalloutThickness</td><td>No. of points</td></tr> </table>	<u>option</u>	<u>value</u>	Opaque	True or False	Callout	True or False	Printable	True or False	Font	Courier New		Arial		etc.	FontFamily	Swiss		Roman		etc.	FontSize	in units eg. 9.0	FontBold	1 or True	FontItalic	1 or True	FontUnderline	1 or True	Spacing	0=Single Point size	Border	1 or True	TextColor	RGB value eg. (255,0,255)	BackColor	RGB value	CalloutColor	RGB value	CalloutThickness	No. of points
<u>option</u>	<u>value</u>																																								
Opaque	True or False																																								
Callout	True or False																																								
Printable	True or False																																								
Font	Courier New																																								
	Arial																																								
	etc.																																								
FontFamily	Swiss																																								
	Roman																																								
	etc.																																								
FontSize	in units eg. 9.0																																								
FontBold	1 or True																																								
FontItalic	1 or True																																								
FontUnderline	1 or True																																								
Spacing	0=Single Point size																																								
Border	1 or True																																								
TextColor	RGB value eg. (255,0,255)																																								
BackColor	RGB value																																								
CalloutColor	RGB value																																								
CalloutThickness	No. of points																																								

Remarks: None.

Return Values: None.

Example: The following Visual Basic code adds a TypeAnywhere object to the form. The TypeAnywhere object is Opaque with Font Size of 12.

```
Call MmaFill.CreateNote(True, "Opaque=True;FontSize=12")
```

The following Visual Basic code adds a TypeAnywhere object with default properties.

```
Call MmaFill.CreateNote(True, "")
```

The following Visual Basic code adds a Sticky Note object to the form. The Sticky Note object is Opaque with Font Size of 12.

```
Call MmaFill.CreateNote(False, "Opaque=True;FontSize=12")
```

The following Visual Basic code adds a Sticky Note object with default properties.

```
Call MmaFill.CreateNote(False, "")
```

Cut

Description:	Moves a marked block of text to the Windows clipboard and deletes it from the field.
Syntax:	C++ <code>BOOL CMmaFill::Cut()</code> Visual Basic <code>[form.]MmaFill.Cut()</code>
Parameters:	None.
Remarks:	None.
Return Values:	True - Call to this method was Successful False - An Error was encountered

DisableRedraw

Description: Disables or enables repainting of the form when field values are being updated via SetFieldData or AppendField methods.

Syntax: C++ `BOOL CMmaFill::DisableRedraw(BOOL Disable)`
 Visual Basic `[form.]MmaFill.DisableRedraw(Byval Disable As Boolean)`

Parameters: The following parameters are available:

Parameter	Description
Disable	set to True to disable redraw/repaint and False otherwise.

Remarks: This method is primarily used in situations where a great number of fields on the form are being updated together (e.g., data from a large database table is being loaded onto the form fields). In order to avoid a redraw on each field update, you can disable redrawing in the beginning of the data-load process and re-enable redraw once the data-load process is complete.

Return Values: True - Call to this method was Successful
 False - An Error was encountered

Example: 'Copy the first 5 characters of LAST_NAME field and Paste it into

'LAST_NAME_2 field

'Use DisableRedraw to eliminate flickering, if any

'Enable Redraw after the procedure

Dim rv

rv = MmaFill1.OpenFormDialog()

rv = MmaFill1.DisableRedraw(True)

rv = MmaFill1.GotoField("LAST_NAME")

rv = MmaFill1.SetCursorPosition(0, 5)

rv = MmaFill1.Copy()

rv = MmaFill1.GotoField("LAST_NAME_2")

rv = MmaFill1.Paste()

rv = MmaFill1.DisableRedraw(False)

DropListAddString

Description: Appends string ChoiceString along with corresponding ChoiceValue to the end of DropList field *FieldName*.

Syntax:

```
C++          long CMmaFill::DropListAddString(LPCTSTR FieldName,
                                                LPCTSTR ChoiceString, LPCTSTR ChoiceValue)

Visual Basic [form.]MmaFill.DropListAddString(Byval FieldName As String,
                                                Byval ChoiceString As String, Byval ChoiceValue As String)
```

Parameter: The following parameters are available

Parameter	Description
Field Name	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName Parameter of this method.
ChoiceString	Choice String to be appended to the end of DropList
ChoiceValue	Choice Value corresponding the ChoiceString

Remarks: None.

Return Values: The zero-based index to the string in the DropList if call to this method was Successful
-1 in case an error was encountered

Example:

```
'Add new entries to DROP_LIST field
'First: Clear all entries in DROP_LIST field
'Second: Add new entries
'Third: Display how many entries we have added to the DROP_LIST field
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.DropListClear("DROP_LIST")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Married", "Married")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Divorced", "Married")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Widowed",
"Widowed")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Single", "Single")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Separated", "Seprated")
```

```
MsgBox "Field 'DROP_LIST' has " +  
MmaFill1.DropListGetCount("DROP_LIST") + " entries."
```

DropListClear

Description: Clear the current selection and all the entries in DropList field FieldName.

Syntax: C++ long CMmaFill::DropListClear(LPCTSTR FieldName)
 Visual Basic [form.]MmaFill.DropListClear(Byval FieldName As String)

Parameters: The following parameters are available

Parameter	Description
Field Name	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName Parameter of this method.

Remarks: None.

Return Values: True - Call to this method was Successful
 False - An Error was encountered

Example: 'Add new entries to DROP_LIST field
 'First: Clear all entries in DROP_LIST field
 'Second: Add new entries
 'Third: Display how many entries we have added to the DROP_LIST field

```
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.DropListClear("DROP_LIST")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Married", "Married")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Divorced", "Married")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Widowed",
"Widowed")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Single", "Single")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Separated",
"Separated")
MsgBox "Field 'DROP_LIST' has " +
MmaFill1.DropListGetCount("DROP_LIST") +
"entries."
```

DropListDeleteString

Description: Deletes a string at the zero-based index nIndex of DropDownList field FieldName.

Syntax: C++ long CMmaFill::DropListDeleteString(LPCTSTR FieldName, short nIndex)

```
Visual Basic [form.]MmaFill.DropListDeleteString(Byval FieldName As  
String, Byval nIndex As Integer)
```

Parameters: The following parameters are available

Parameter	Description
Field Name	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.
nIndex	Specified the zero-based index to the string to be deleted

Remarks: None.

Return Values: Returns a count of strings remaining in the DropDownList if call to this method was Successful.

Returns -1 in case an error was encountered.

Example: 'Field "Marital_Status" is a droplist object and has these choices: Married, Single, Divorced, Widowed

 $\dim \mathbf{RV}$

```
RV = DropListDeleteString ("Marital status",2)
```

‘After this command is executed, field "Marital_Status" is left with these choices: Married, Single, Widowed

"Divorced", which is the 2nd choice (Married is 0th choice, Single is 1st choice and Divorced is 2nd choice) is deleted.

'RV contains the return value from this command, which is the number of remaining choices, which will be 3 after the command is executed. therefore, RV will be set to 3.

DropListGetCount

Description: Returns the number of items in the DropList field FieldName.

Syntax: C++ long CMmaFill::DropListGetCount(LPCTSTR FieldName)
 Visual Basic [form.]MmaFill.DropListGetCount(Byval FieldName As String)

Parameters: The following parameters are available

Parameter	Description
Field Name	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName Parameter of this method.

Remarks: None.

Return Values: Returns the number of items in the DropList if call to this method was Successful

Returns -1 in case an error was encountered

Example:

```
'Add new entries to DROP_LIST field
'First: Clear all entries in DROP_LIST field
'Second: Add new entries
'Third: Display how many entries we have added to the DROP_LIST field
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.DropListClear("DROP_LIST")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Married", "Married")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Divorced", "Married")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Widowed", "Widowed")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Single", "Single")
rv = MmaFill1.DropListAddString("DROP_LIST", "I am Separated",
"Separated")
MsgBox "Field 'DROP_LIST' has " +
MmaFill1.DropListGetCount("DROP_LIST") + " entries."
```

DropListGetCurSel

Description: Returns the zero-based index of the currently selected item in the DropListfield FieldName.

Syntax: C++ long CMmaFill::DropListGetCurSel(LPCTSTR FieldName)
Visual Basic [form.]MmaFill.DropListGetCurSel(Byval FieldName As String)

Parameters: The following parameters are available

Parameter	Description
Field Name	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName Paramater of this method.

Remarks: None.

Return Values: Returns the zero-based index of the currently selected item if call to this method was successful

Returns -1 in case an error was encountered

Example: 'Field "Marital_Status" is a droplist object and has these choices: Married, Single, Divorced, Widowed
dim RV
RV = DropListGetCurSel ("Marital_status")
'If user selects "Divorced," then this command will return 2 (i.e. RV is set to 2), which is the 2nd choice (Married is 0th choice, Single is 1st choice and Divorced is 2nd choice) is deleted.
'If there is an error, then this command returns -1 (i.e. RV is set to -1).

DropListSetCurSel

Description: Selects a string in the DropList field FieldName.

Syntax: C++ long CMmaFill::DropListSetCurSel(LPCTSTR FieldName, short nSelect)

```
Visual Basic [form.]MmaFill.DropListSetCurSel(Byval FieldName As String,  
Byval nSelect As Integer)
```

Parameters: The following parameters are available

Parameter	Description
Field Name	Name or TabOrder of the DropDownList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.
nSelect	Specifies the zero-based index of the string to select. If -1, any current selection in the DropDownList is removed and the field is cleared

Remarks: None.

Return Values: Returns the zero-based index of the item selected if call to this method was Successful

Returns -1 in case an error was encountered

Example: 'Field "Marital_Status" is a droplist object and has these choices: Married, Single, Divorced, Widowed

 $\dim RV$

RV = DropDownListCurSel ("Marital status",3)

‘After this command is executed, "Widowed" will display in the droplist field as the user's choice. Note that "Widowed" is the 3rd choice in the droplist.

‘If there is an error, then this command returns -1 (i.e. RV is set to -1).

EnableAddendumTag

Description: Disables or enables appearance of the words “Addendum Tags” for fields whose text exceed their size. This command applies to all fields on the form.

Syntax: C++ `BOOL CMmaFill::EnableAddendumTag(BOOL Enable)`
Visual Basic `[form.]MmaFill.EnableAddendumTag(Byval Enable As Boolean)`

Parameters: The following parameters are available

Parameter	Description
Enable	set to True to enable drawing of Addendum Tags and False otherwise.

Remarks: When EnableAddendumTag is invoked, any field on the form with its text exceeding its size will be tagged with “See Addendum...” in its lower-right corner. At print time, you can give the option of printing an Addendum page containing a cross-reference to the Addendum Tags and the spill-over text for the corresponding fields to the user.

Once this method is called, the changes remain in effect for subsequent forms loaded into the control until the next call to this method.

When this is not invoked, the addendum is still there but there is no visual cue on the screen with the text “See Addendum.”

Return Values: True - Call to this method was Successful

False - An Error was encountered

Example: 'Following shows how to Enable Addendum Tags for ONLY the LAST_NAME

'field

'First: Disable ALL Addendum Tags

'Second: Enable Addendum Tag for ONLY the LAST_NAME field

Dim rv

rv = MmaFill1.OpenFormDialog()

rv = MmaFill1.EnableAddendumTag(False)

rv = MmaFill1.EnableFieldAddendumTag("LAST_NAME", True)

EnableField

Description: Disables or enables overwrite property of the field FieldName.

Syntax:

C++ `BOOL CMmaFill::EnableField(LPCTSTR FieldName, BOOL Enable)`

Visual Basic `[form.]MmaFill.EnableField(Byval FieldName As String, Byval Enable As Boolean)`

Parameters: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.
Enable	set to True to enable overwrite for field FieldName and False otherwise

Remarks: If you attempt to disable a field that currently has the focus, the field will be disabled and focus will be given to the next field in tab order.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

'Following shows how to ENABLE the LAST_NAME field on the form ONLY

'First: Disable ALL fields on the form

'Second: Enable the LAST_NAME field

```
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.EnableFields(0, False)
rv = MmaFill1.EnableField("LAST_NAME", True)
```

EnableFieldAddendumTag

Description: Disables or enables appearance of the words “Addendum Tags” for FieldName.

Syntax:

```
C++          BOOL CMmaFill::EnableFieldAddendumTag(LPCTSTR
              FieldName, BOOL Enable)

Visual Basic [form.]MmaFill.EnableFieldAddendumTag(Byval FieldName As
              String, Byval Enable As Boolean)
```

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.
Enable	set to True to enable overwrite for field FieldName and False otherwise

Remarks: When EnableFieldAddendumTag is invoked for FieldName and text of FieldName exceeds its size, the field will be tagged with “See Addendum ...” in its lower-right corner.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

```
'Following shows how to Enable Addendum Tags for ONLY the LAST_NAME
'field

'First: Disable ALL Addendum Tags

'Second: Enable Addendum Tag for ONLY the LAST_NAME field

Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.EnableAddendumTag(False)
rv = MmaFill1.EnableFieldAddendumTag("LAST_NAME", True)
```

EnableFields

Description: Disables or enables overwrite property of all the fields on page PageNum. If PageNum is zero, then disabling/enabling process is applied to all of the fields on all of the pages on the currently loaded form.

Syntax: C++ `BOOL CMmaFill::EnableFields(short PageNum, BOOL Enable)`
Visual Basic `[form.]MmaFill.EnableFields(Byval PageNum As Integer, Byval Enable As Boolean)`

Parameter: The following parameters are available

Parameter	Description
PageNum	Page number of the form
Enable	set to True to enable overwrite for fields on page PageNum and False otherwise.

Remarks: At the completion of the call to this method, the focus will be set to the first field of first page.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Disable ALL fields on the form
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.EnableFields(0, False)

'Disable ALL fields on page 1 of the form
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.EnableFields(1, False)

FileDialog

Description:	The FileDialog provides a standard dialog box for operations such as opening and saving files.		
Syntax:	C++	VARIANT CMmaFill:: FileDialog (BOOL bOpenFileDialog, LPCTSTR Filter, LPCTSTR Options)	
	Visual Basic	[form.]MmaFill. FileDialog (bOpenFileDialog As Boolean, Filter As String, Options As String)	

Parameter: The following parameters are available

Parameter	Description
bOpenFileDialog	Show the OpenFile Dialog box
Filter	<p>The Filter parameter Syntax: has these parts:</p> <ul style="list-style-type: none"> • Object: An object expression that evaluates to an object in the “Applies To” list. • Description: A string expression describing the type of file. • Filter: A string expression specifying the filename extension.
Options	<p>The Following parameter/value pairs are separated by a semicolon:</p> <ul style="list-style-type: none"> • "InitialDir=<Initial Directory>" FileDialog will initially show the contents of <Initial Directory> • "Title=<Dialog Caption>" Sets FileDialog caption to <Dialog Caption>

Remarks: A Filter specifies the type of files that are displayed in the dialog box's file list box. For example, selecting the filter *.txt displays all text files.

Use this property to provide the user with a list of filters that can be selected when the dialog box is displayed.

Use the pipe (|) symbol (ASCII 124) to separate the description and filter values. Do not include spaces before or after the pipe symbol, because these spaces will be displayed with the description and filter values.

The following code shows an example of a filter that enables the user to select text files or graphic files that include bitmaps and icons:

```
Text (*.txt)|*.txt|Pictures (*.bmp;*.ico)|*.bmp;*.ico|
```


When you specify more than one filter for a dialog box, use the Filter Index property to determine which filter is displayed as the default.

Note that “||” terminates the “Filter” string.

Example:

'Open a form using the Generic FileDialog method

'First: Use FileDialog() to get the name of the form

'Second: Use OpenForm() to the actual opening of the form

```
Dim FileName As String
```

```
Dim rv
```

```
FileName = FormFiller.FileDialog(True, "Visual eForms (*.far)|*.far|Data Files  
(*.dat)|*.dat|All (*.*)|*.*||", "INITIALDIR=c:\temp;TITLE=My Title")
```

```
rv = MmaFill1.OpenForm(FileName)
```

FillTestData

Description:	Fills all the fields on the currently loaded form with test data.
Syntax:	C++ <code>BOOL CMmaFill::FillTestData()</code> Visual Basic <code>[form.]MmaFill.FillTestData()</code>
Parameters:	None.
Remarks:	This method could be used to examine/evaluate the look-and-feel of a filled form.
Return Values:	True - Call to this method was Successful False - An Error was encountered
Example:	'First: Fill form fields with test data 'Second: Clear data from all fields on the form Dim rv rv = MmaFill1.OpenFormDialog() rv = MmaFill1.FillTestData() rv = MmaFill1.ClearData()

GetCurrField

Description: Sets FieldName to the name of the current field in focus.

Syntax: C++ BOOL CMmaFill::GetCurrField(BSTR FAR* FieldName)
Visual Basic [form.]MmaFill.GetCurrField(FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Holder for the name of the form Field in focus

Remarks: none

See Also: VarGetCurrField() - for VB or Java scripting.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Two ways of capturing the name of field that has Focus:
'GetCurrField() and VarGetCurrField()
Dim rv
Dim FieldName As String * 255
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.GetCurrField(FieldName)
MsgBox RTrim(FieldName) + " has now focus."
MsgBox MmaFill1.VarGetCurrField() + " has now focus."

GetCurrPage

Description: Returns the current page number of the currently loaded form.

Syntax: C++ short CMmaFill::GetCurrPage()
Visual Basic [form.]MmaFill.GetCurrPage()

Parameters: None.

Remarks: None.

Return Values: Current page number of the currently loaded form.

Example: 'What page am I on now?
'How many fields does this form have?
Dim rv
Dim PageNum, FieldCount
rv = MmaFill1.OpenFormDialog()
PageNum = MmaFill1.GetCurrPage()
FieldCount = MmaFill1.GetFieldCount()
MsgBox "You are currently on page #" & PageNum & " and there are " &
FieldCount & " fields on this form"

GetFieldAddendumLen

Description: Returns the length of field FieldName's Addendum (i.e., number of characters in Addendum part of field FieldName).

Syntax:

```
C++          short CMmaFill::GetFieldAddendumLen(LPCTSTR FieldName)
Visual Basic [form.]MmaFill.GetFieldAddendumLen(Byval FieldName As
String)
```

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName Paramater of this method.

Remarks: None.

Return Values: Short - Length of field FieldName's Addendum.

Example:

```
dim rv
rv = mmafill1.GetFieldAddendumLen ("Description")
```

'After this command is executed, "rv" will contain the number of charaters that are placed into the addendum for field "Description".

GetFieldAddendumText

Description: Assigns Addendum Text of field FieldName to AddendumText.

Syntax:

```
C++          BOOL CMmaFill::GetFieldAddendumText(LPCTSTR FieldName,
          BSTR FAR* AddendumText)

Visual Basic [form.]MmaFill.EnableFieldAddendumTag(Byval FieldName As
          String, AddendumText As String)
```

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field
AddendumText	Holder for Addendum Text of field FieldName

Remarks: Addendum text of a field refers to excess text that cannot be fit within the bounds of that field.

See Also: VarGetFieldAddendumText()

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

```
'We have two forms: AnyForm.far and AddendumForm.far
'We want to open, fill and print the AnyForm.far.
'We also want to print the excess data that user has typed
'into the fields of AnyForm.far to be printed via our
'addendum form (AddendumForm.far).
'for this to happen we add two instance of Visual eForms ActiveX
'to our Visual Basic project:
'MMAFill1 - Will hold the original form (AnyForm.far),
'and is accessible to the user
'AddendumFiller - Will house the Addendum form (AddendumForm.far),
'and is NOT visible to the user.
'

Dim rv, i, FormFieldName As String, AddendumFieldName As String
Dim AddendumCounter As Integer, FieldAddendumText As String
```

```
Dim FormAddendumText As String
Dim PrinterDC, TotalPages As Integer
',

'Open Form - This routine opens "AnyForm.far"
'You could also use OpenFileDialog() to allow user to select and
'then open a form
'This routine also sets the "Overwrite" property of all fields in this form
'to TRUE.
',

rv = MmaFill1.OpenForm(App.Path & "\AnyForm.far")
rv = MmaFill1.EnableFields(0, True)
'FieldAddendumText - holds the addendum for a field in AnyForm
'FormAddendumText - holds the addendum for all fields in the AnyForm
',

'FormFieldName - name of a field in AnyForm
'AddendumFieldName - name of the addendum field in AddendumForm
'Print Form - This routine prints the original form (AnyForm.far) and then
'moves the addendum data for all fields to a separate form (AddendumForm.far)
',
',

'GetFirstField - Returns the name of the first field in the Tabbing order
',

FormFieldName = MmaFill1.GetFirstField
',

'Print the AnyForm
'We will ask the user to select a printer using PrintGetDC() method.
'PrintGetDC() method returns the Device Context (DC) of the selected printer.
'We then call PrintForm() method to print from page 1 to page N
'of the AnyForm
',

PrinterDC = MmaFill1.PrintGetDC("")
If PrinterDC = 0 Then Exit Sub
'user selected the CANCEL button. Exit
TotalPages = MmaFill1.GetNumPages()
rv = MmaFill1.PrintForm(PrinterDC, 1, TotalPages)
```

```

',
'We will now traverse (using GetNextField() method) through all of the
'Fillable Fields of AnyForm.
'As we land on each Fillable Field, we examine it for addendum data
'We build an addendum string by adding all of these field-based addendum data.
'We will place all addendum data for all fields of AnyForm into
'FormAddendumText
'Construct the whole addendum text for the AnyForm
',

AddendumCounter = 1
FormAddendumText = ""
For i = 1 To MmaFill1.GetFieldCount
    DoEvents
    FieldAddendumText = MmaFill1.VarGetFieldAdden -
dumText(FormFieldName)
    If FieldAddendumText <> "" Then
        FormAddendumText = FormAddendumText & "This is _
Addendum " & AddendumCounter & Chr(13) & Chr(10)
        FormAddendumText = FormAddendumText & FieldAd _
dendumText & Chr(13) & Chr(10) & Chr(13) & Chr(10)
        AddendumCounter = AddendumCounter + 1
    End If
    FormFieldName = MmaFill1.GetNextField
Next i
',
'We now have traversed through all the Fillable fields.
'It is time to print the AddendumForm
',
'We collect all addendum text for all fields of AnyForm, and then
'paste & print it into the AddendumForm one chunk at a time until
'all Addendum data is processed properly.
',
',
'Open the Addendum form into a separate container

```



```

'(another instance of Filler component)
'Set the Addendum Tag for all fields of this form to FASLE
'
rv = AddendumFiller.OpenForm(App.Path & "\AddendumForm.far")
rv = AddendumFiller.EnableAddendumTag(False)
'
'VarGetCurrField - returns the name of the field that has the focus
'(i.e., the addendum field)
'
AddendumFieldName = AddendumFiller.VarGetCurrField()
'
'1. Paste the addendum text we collected from AnyForm into the
'AddendumForm
'2. Print the form.
'3. Check to see if there is any more addendum text left
'4. if so, then go to step 1
'5. continue until all addendum text is processed and printed.
'
If FormAddendumText <> "" Then
    Do
        DoEvents
        rv = AddendumFiller.SetFieldData(AddendumFileName, FormAd _
            dendumText)
        rv = AddendumFiller.PrintForm(PrinterDC, 1, Addendum _
            Filler.GetNumPages())
        FormAddendumText = AddendumFiller.VarGetField _
            AddendumText(AddendumFileName)
    Loop Until FormAddendumText = ""
End If
'
'NOTE: Always FREE the Device Context (DC) using PrintFreeDC() method.
'
MmaFill1.PrintFreeDC

```

GetFieldCount

Description: Returns the total number of fields on the currently loaded form.

Syntax: C++ short CMmaFill::GetFieldCount()
Visual Basic [form.]MmaFill.GetFieldCount()

Parameters: None.

Remarks: None.

Return Values: Short - Total number of fields on the currently loaded form.

Example: 'What page am I on now?
'How many fields does this form have?
Dim rv
Dim PageNum, FieldCount
rv = MmaFill1.OpenFormDialog()
PageNum = MmaFill1.GetCurrPage()
FieldCount = MmaFill1.GetFieldCount()
MsgBox "You are currently on page #" & PageNum & " and there are " &
FieldCount & " fields on this form"

GetFieldHelp

Description: Assigns Help Text of field FieldName to FieldHelp.

Syntax:

C++ BOOL CMmaFill::GetFieldHelp(LPCTSTR FieldName, BSTR FAR* FieldHelp)

Visual Basic [form.]MmaFill.GetFieldHelp(Byval FieldName As String, FieldHelp As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field
FieldHelp	Holder for Help Text of field FieldName

Remarks: Help Text for form fields are set at Design time by assigning appropriate text to Edit/FieldHelp property of form fields.

See Also: VarGetFieldHelp()

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example:

```
'Display the "FieldHelp" for the field that the user is in now
'Two methods can be used: GetFieldHelp() and VarGetFieldHelp()
Dim rv
Dim FieldHelp As String * 256
Dim CurrentField As String
rv = MmaFill1.OpenFormDialog()
CurrentField = MmaFill1.VarGetCurrField()
rv = MmaFill1.GetFieldHelp(CurrentField, FieldHelp)
MsgBox "FieldHelp is (using methid #1): " & RTrim(FieldHelp)
MsgBox "FieldHelp is (using methid #2): " & MmaFill1.VarGetFieldHelp()
```

GetFieldLen

Description Returns the length of field FieldName's data (i.e., number of characters entered into field FieldName).

Syntax: C++ short CMmaFill::GetFieldLen(LPCTSTR FieldName)
Visual Basic [form.]MmaFill.GetFieldLen(Byval FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the Form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName parameter of this method.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'How much text have I typed into this field?
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.AppendField("LAST_NAME", "Add this Text to the end of Last_Name field")
rv = MmaFill1.GetFieldLen("LAST_NAME")
MsgBox "LAST_NAME now has " & rv & " characters in it"

GetFieldLineCount

Description: Returns the number of lines of text in field FieldName's data.

Syntax: C++ short CMmaFill::GetFieldLineCount(LPCTSTR FieldName)
Visual Basic [form.]MmaFill.GetFieldLineCount(Byval FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with "@"; for example, to address a field at Tab Order 12, pass "@12" as FieldName Paramater of this method.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

GetFieldLong

Description: Assigns the value of field FieldName to FieldData.

Syntax:

C++ `BOOL CMmaFill::GetFieldLong(LPCTSTR FieldName, long FAR* FieldData)`

Visual Basic `[form.]MmaFill.GetFieldLong(Byval FieldName As String, FieldData As Long)`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the Form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName parameter of this method.
FieldData	Holder for value of field FieldName

Remarks: Since this method coerces the value of field FieldName to a Long integer, the method should be primarily used for numeric or checkmark fields.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

GetFieldProperty

Description: Returns the field property value corresponding to PropertyID.

Syntax:

C++ BSTR CMmaFill::GetFieldProperty(LPCTSTR FieldName, long PropertyID)

Visual Basic [form.]MmaFill.GetFieldProperty(Byval FieldName As String, Byval PropertyID As Long)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field
PropertyID	ID of the property whose value is of interest (See below for valid IDs)

Remarks: None.

Return Values: Valid Property IDs and their Return values:

Property	ID	Returns																						
Type	1	Number-String representing the Type of the object represented by 'FieldName'. Possible Values are: <table><tr><td>8</td><td>vfFillableText</td></tr><tr><td>16</td><td>vfCheckBox</td></tr><tr><td>32</td><td>vfButton</td></tr><tr><td>256</td><td>vfNumber</td></tr><tr><td>512</td><td>vfEditableImage</td></tr><tr><td>2048</td><td>vfDate</td></tr><tr><td>16384</td><td>vfBarCode</td></tr><tr><td>65536</td><td>vfMask</td></tr><tr><td>131072</td><td>vfDropList</td></tr><tr><td>262144</td><td>vfSignature</td></tr><tr><td>2097152</td><td>vfHyperlink</td></tr></table>	8	vfFillableText	16	vfCheckBox	32	vfButton	256	vfNumber	512	vfEditableImage	2048	vfDate	16384	vfBarCode	65536	vfMask	131072	vfDropList	262144	vfSignature	2097152	vfHyperlink
8	vfFillableText																							
16	vfCheckBox																							
32	vfButton																							
256	vfNumber																							
512	vfEditableImage																							
2048	vfDate																							
16384	vfBarCode																							
65536	vfMask																							
131072	vfDropList																							
262144	vfSignature																							
2097152	vfHyperlink																							
Line Width	7	Returns the LineThickness property of the field.																						

Property	ID	Returns
Back Color	8	String representing the RGB value of the Color (e.g., "255,0,0" for Red)
Line Color	9	String representing the RGB value of the Color (e.g., "0,255,0" for Green).
Left Border	10	"1" if the Field has its Left Border enabled and "0" otherwise.
Top Border	11	"1" if the Field has its Top Border enabled and "0" otherwise.
Right Border	12	"1" if the Field has its Right Border enabled and "0" otherwise.
Bottom Border	13	"1" if the Field has its Bottom Border enabled and "0" otherwise.
Rounded Border	14	"1" if the Field has Rounded Borders and "0" otherwise.
Opaque	15	Returns "1" if the Opaque property is TRUE. Otherwise, Returns "0"
Non-Printable	16	Returns "1" if the NonPrintable property is TRUE. Otherwise, Returns "0"
Repeat On All Pages	17	Returns "1" if the RepeatAllPages property is TRUE. Otherwise, Returns "0"
Text Orientation	18	Returns the value of the Orientation property of a field.
Left Margin	19	Returns the value of the Left Margin assigned to the field.
Top Margin	20	Returns the value of the Top Margin assigned to the field.
Right Margin	21	Returns the value of the Right Margin assigned to the field.
Bottom Margin	22	Returns the value of the Bottom Margin assigned to the field.
Line Spacing	23	Returns the Spacing property assigned to a field.

Property	ID	Returns
Horizontal Justification	24	Returns the value of the JustHorz property of a field.
Vertical Justification	25	Returns the value of the JustVert property of a field.
Text Color	26	String representing the RGB value of the Color (e.g., "0,0,255" for Blue).
Max Fill Chars	27	Number-String representing the Maximum Fillable Characters (e.g., "25").
Overwrite	28	"1" if the Field is Enabled and "0" otherwise.
Notify Click	29	Returns "1" if the Click Notify event is "ON" Otherwise, Returns "0"
Notify Modify	30	Returns "1" if the Modify Notify event is "ON" Otherwise, Returns "0"
Notify DblClick	31	Returns "1" if the DblClick Notify event is "ON" Otherwise, Returns "0"
Notify Got Focus	32	Returns "1" if the GotFocus Notify event is "ON" Otherwise, Returns "0"
Notify Lost Focus	33	Returns "1" if the LostFocus Notify event is "ON". Otherwise, Returns "0"
Notify Mouse Enter	34	Returns "1" if the MouseEnter Notify event is "ON" Otherwise, Returns "0"
Notify Mouse Exit	35	Returns "1" if the MouseExit Notify event is "ON" Otherwise, Returns "0"
TAB Order	36	Returns the value of the TabOrder property of a field.
ON String	37	Returns the value of the StrOn property of a CheckBox field.
OFF String	38	Returns the value of the StrOff property of a CheckBox field.
EMPTY String	39	Returns the value of the StrEmpty property of a CheckBox field.
Page Number	40	Number-String representing the Page Number a given field appears on.

Property	ID	Returns
Calculation	41	Returns the value of the Calculation property of a field.
Fill Font	43	<p>The string describing a Fill Font has the following format:</p> <p>"name,height,bold,italic,underline,char-spacing mode, char-spacing points"</p> <p>where</p> <p><i>name</i></p> <p>font name</p> <p><i>height</i></p> <p>font size in points</p> <p><i>bold</i></p> <p>True or False</p> <p><i>italic</i></p> <p>True or False</p> <p><i>underline</i></p> <p>True or False</p> <p><i>char-spacing mode</i></p> <p>Normal=0</p> <p>Expanded=1</p> <p>Condensed=2</p> <p>Fixed=3</p> <p><i>char-spacing points</i></p> <p>increments of 1/10th of point applied to 'Expanded', 'Condensed' or 'Fixed' mode.</p>
Field Help	44	Returns the value of the FiedHelp property of a field.
Date Format	45	Returns the value of the DateFormat property of a DATE field.
Date Auto	46	Returns "1" if the DateAuto property of a DATE field is TRUE. Otherwise, Returns "0"
Decimal Points	47	Returns the value assigned to DecPoints property of a Number or Currency field.

Property	ID	Returns
Decimal Character	48	Returns the character assigned to DecChar property of a Number or Currency field.
Comma Separated	49	Returns "1" if the CommaSep property of a Number or Currency field is TRUE. Otherwise, Returns "0".
Imeg File	50	Returns the value of the ImageFile property of an Image field.
Scale Type	51	Returns the value assigned to ScaleType property of an Image field.
Scale_X	52	Returns the value assigned to ScaleX property of an Image field.
Scale_Y	53	Returns the value assigned to ScaleY property of an Image field.
Crop_X	54	Returns the value assigned to CropX property of an Image field.
Crop_Y	55	Returns the value assigned to CropY property of an Image field.
Fill Char	57	Returns the value assigned to FillChar property of a field.
Symbology	58	Returns the value assigned to Symbology property of a Bar Code field.
Bar Width	59	Returns the BarWidth property of a Bar Code field.
Ratio	60	Returns the value assigned to Ratio property of a Bar Code field.
Checksum	61	Returns "1" if the CheckSum property of a Bar Code field is TRUE. Otherwise, Returns "0"
Caption Align	62	Returns the value assigned to CaptionAlign property of a Bar Code field.
Value	63	Returns the DefaultValue of a field.
Mandatory	65	Returns "1" if the Mandatory property of a field is TRUE. Otherwise, Returns "0"
Auto Tab	66	Returns "1" if the AutoTab property of a field is TRUE. Otherwise, Returns "0"
Left Arrow	67	Returns "1" if the ArrowLeft property of a Line field is TRUE. Otherwise, Returns "0"

Property	ID	Returns
Right Arrow	68	Returns "1" if the ArrowRight property of a Line field is TRUE. Otherwise, Returns "0"
Pen Style	69	Returns the value assigned to PenStyle property of a field.
Mask	70	Returns the Mask property of a Mask field.
User Modify	71	Returns "1" if the UserModify property of a Bar Code field is TRUE. Otherwise, Returns "0"
List Height	72	Returns the ListHeight property of a Drop List field.
List Width	73	Returns the ListWidth property of a Drop List field.
List Choice	74	Returns a list of choices/values assigned to the List property of a Drop List field.
Embedded	75	Returns "1" if the Embedded property of an Image field is TRUE. Otherwise, Returns "0"
Visible	76	"1" if the Field is Visible and "0" otherwise.
Signature Type	77	Returns the value assigned to Type property of a Signature field.
Signature Mode	78	Returns the value assigned to Mode property of a Signature field.
Signature Dependant Fields	79	Returns the list of fields assigned to Form Fields property of a Signature field.
Signature Lock Fields	80	Returns "1" if the Lock Fields property of a Signature field is TRUE. Otherwise, returns "0"

Example:

```
'What are the properties of this field?
'NOTE: if NONE of the objects on the form are
'Enabled (overwrite=true)
'then the GetFieldProperty() routine will fail.
'

Dim rv

On Error Resume Next

rv = MmaFill1.OpenFormDialog()
'what is the background color of the field in focus?
rv = MmaFill1.GetFieldProperty(MmaFill1.VarGetCurrField, MmaFill1.vfBack-
Color)
MsgBox "The RGB value of the BackColor is: " & rv
'what type of object is the field that has the focus?
rv = MmaFill1.GetFieldProperty(MmaFill1.VarGetCurrField, MmaFill1.vfType)
Select Case rv
Case MmaFill1.vfBarCode:

MsgBox "object is 'BARCODE'"

Case MmaFill1.vfButton:

MsgBox "object is 'BUTTON'"

Case MmaFill1.vfCheckBox:

MsgBox "object is 'CHECKBOX'"

Case MmaFill1.vfDate:

MsgBox "object is 'DATE'"

Case MmaFill1.vfDropList:
```

```
MsgBox "object is 'DROPLIST'"
```

```
Case MmaFill1.vfEditableImage:
```

```
MsgBox "object is 'EDITABLE IMAGE'"
```

```
Case MmaFill1.vfFillableText:
```

```
MsgBox "object is 'FILLABLE FIELD'"
```

```
Case Else
```

```
MsgBox "object is UNKNOWN"
```

```
End Select
```

```
'is the TOP border of the field in focus ON or OFF?
```

```
rv = MmaFill1.GetFieldProperty(MmaFill1.VarGetCurrField, MmaFill1.vfTop-  
Border)
```

```
If rv = "1" Then
```

```
MsgBox "TOP border is ON"
```

```
Else
```

```
MsgBox "TOP border is OFF"
```

```
End If
```

GetFieldString

Description: Assigns the value of field FieldName to FieldData.

Syntax:

C++ `BOOL CMmaFill::GetFieldString(LPCTSTR FieldName, BSTR FAR* FieldData)`

Visual Basic `[form.]MmaFill.GetFieldString(Byval FieldName As String, FieldData As String)`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the Form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName parameter of this method.
FieldData	Holder for value of field FieldName

Remarks: None.

See Also: VarGetFieldString()

Return Values:

True - Call to this method was Successful

False - An Error was encountered

GetFieldTextWidth

Description: Returns the width of Text according to the Font attributes of the field
FieldName.

Syntax:	C++	long CMmaFill::GetFieldTextWidth(LPCTSTR FieldName, LPCTSTR Text)
	Visual Basic	[form.]MmaFill.GetFieldTextWidth(Byval FieldName As String, Byval Text As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.
Text	String of text for which this method calculates its width

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

GetFirstField

Description:	Returns the name of the first field in tabbing order on the form.
Syntax:	C++ VARIANT CMmaFill::GetFirstField() Visual Basic [form.]MmaFill.GetFirstField()
Parameter:	None
Remarks:	none
See Also:	See Also GetNextField()
Return Values:	Variant - Name of the first Field Null - An Error was encountered or no fields were found on the form
Example:	<pre> 'We have two forms: AnyForm.far and AddendumForm.far 'We want to open, fill and print the AnyForm.far. 'We also want to print the excess data that user has typed 'into the fields of AnyForm.far to be printed via our 'addendum form (AddendumForm.far). 'for this to happen we add two instance of Visual eForms ActiveX 'to our Visual Basic project: 'MMAFill - Will hold the original form (AnyForm.far), 'and is accessible to the user 'AddendumFiller - Will house the Addendum form (AddendumForm.far), 'and is NOT visible to the user. ' Dim rv, i, FormFieldName As String, AddendumFieldName As String Dim AddendumCounter As Integer, FieldAddendumText As String Dim FormAddendumText As String Dim PrinterDC, TotalPages As Integer ' 'Open Form - This routine opens "AnyForm.far" 'You could also use OpenFormDialog() to allow user to select and 'then open a form </pre>

```
'This routine also sets the "Overwrite" property of all fields in this form
'to TRUE.
,

rv = MmaFill1.OpenForm(App.Path & "\AnyForm.far")
rv = MmaFill1.EnableFields(0, True)

'FieldAddendumText - holds the addendum for a field in AnyForm
'FormAddendumText - holds the addendum for all fields in the AnyForm
,

'FormFieldName - name of a field in AnyForm
'AddendumFieldName - name of the addendum field in AddendumForm
'Print Form - This routine prints the original form (AnyForm.far) and then
'moves the addendum data for all fields to a separate form (AddendumForm.far)
,
,

'GetFirstField - Returns the name of the first field in the Tabbing order
,

FormFieldName = MmaFill1.GetFirstField
,

'Print the AnyForm
'We will ask the user to select a printer using PrintGetDC() method.
'PrintGetDC() method returns the Device Context (DC) of the selected printer.
'We then call PrintForm() method to print from page 1 to page N
'of the AnyForm
,

PrinterDC = MmaFill1.PrintGetDC("")
If PrinterDC = 0 Then Exit Sub
'user selected the CANCEL button. Exit
TotalPages = MmaFill1.GetNumPages()
rv = MmaFill1.PrintForm(PrinterDC, 1, TotalPages)
,

'We will now traverse (using GetNextField() method) through all of the
'Fillable Fields of AnyForm.
'As we land on each Fillable Field, we examine it for addendum data
'We build an addendum string by adding all of these field-based addendum data.
'We will place all addendum data for all fields of AnyForm into
```

```
'FormAddendumText
'Construct the whole addendum text for the AnyForm
'
AddendumCounter = 1
FormAddendumText = ""
For i = 1 To MmaFill1.GetFieldCount
    DoEvents
    FieldAddendumText = MmaFill1.VarGetFieldAdden -
dumText(FormFieldName)
    If FieldAddendumText <> "" Then
        FormAddendumText = FormAddendumText & "This is _
Addendum " & AddendumCounter & Chr(13) & Chr(10)
        FormAddendumText = FormAddendumText & FieldAd _
dendumText & Chr(13) & Chr(10) & Chr(13) & Chr(10)
        AddendumCounter = AddendumCounter + 1
    End If
    FormFieldName = MmaFill1.GetNextField
Next i
'
'We now have traversed through all the Fillable fields.
'It is time to print the AddendumForm
'
'We collect all addendum text for all fields of AnyForm, and then
'paste & print it into the AddendumForm one chunk at a time until
'all Addendum data is processed properly.
'
'
'Open the Addendum form into a separate container
'(another instance of Filler component)
'Set the Addendum Tag for all fields of this form to FASLE
'
rv = AddendumFiller.OpenForm(App.Path & "\AddendumForm.far")
rv = AddendumFiller.EnableAddendumTag(False)
'
```

```

'VarGetCurrField - returns the name of the field that has the focus
'(i.e., the addendum field)
'
AddendumFieldName = AddendumFiller.VarGetCurrField()
'

'1. Paste the addendum text we collected from AnyForm into the
'AddendumForm
'2. Print the form.
'3. Check to see if there is any more addendum text left
'4. if so, then go to step 1
'5. continue until all addendum text is processed and printed.
'

If FormAddendumText <> "" Then
    Do
        DoEvents
        rv = AddendumFiller.SetFieldData(AddendumFileName, FormAd _
            dendumText)
        rv = AddendumFiller.PrintForm(PrinterDC, 1, Addendum _
            Filler.GetNumPages())
        FormAddendumText = AddendumFiller.VarGetField _
            AddendumText(AddendumFileName)
    Loop Until FormAddendumText = ""
End If
'

'NOTE: Always FREE the Device Context (DC) using PrintFreeDC() method.
'

MmaFill1.PrintFreeDC

```

GetFormPath

Description:	Returns the Form-Path associated with the currently loaded form.
Syntax:	<div>C++ VARIANT CMmaFill::GetFormPath()</div> <div>Visual Basic [form.]MmaFill.PrintGetDC () As Variant</div>
Parameters:	None.
Remarks:	This method can be used to return the full Form-Path associated with the currently loaded form or to test whether there is any form currently loaded.
Return Values:	Form-Path associated with the currently loaded form.
Example	<div>If mmafill1.GetFormPath = "" then</div> <div>MsgBox "Warning: No Form Currently Loaded"</div> <div>End If</div>

GetFormProperty

Description: Returns the form property value corresponding to PropertyID.

Syntax: C++ VARIANT CMmaFill::GetFormProperty(long PropertyID)
 Visual Basic [form.]MmaFill.GetFormProperty(Byval PropertyID As Long)

Parameter: The following parameters are available

Parameter	Description
PropertyID	ID of the property, whose value will be returned by GetFormProperty()

Remarks: None.

Return Values: GetFormProperty returns values of the following properties:

Property	ID	Returns
Form Description	101	String representing the Description: of the currently loaded form as set in the Designer.
Active-Field Color	106	String representing the RGB value of the Color (e.g., "255,0,0" for Red).
Background Color	107	String representing the RGB value of the Color (e.g., "0,255,0" for Green).
Orientation	108	"1" if the Form Orientation is Portrait; "0" if the Form Orientation is Landscape.
TCPIP Progress Window	109	"1" if the Progress Window is to be displayed; "0" otherwise.
Allow Content Search	115	"1"
Track History	116	"1"
Form Width	103	Width of the form in inches.
Form Height	104	Height of the form in inches.
Form Name	100	Get the Name property of the form
Form Version	102	Get version of the form.
Author	112	Get the name of the person who designed the form.
Category	113	Get the category to the form
Search Keywords	114	Get a comma-delimited list of keywords included in the "Search Keyword" property of the form.
Copyright	117	Get the Copyright clause of the form.
Comments	118	Get Comments of the designer.

Property	ID	Returns
Index Fields	119	Get a Comma-delimited list of fields included in the “Index Fields” property of the form.
Archive Format	120	Get the file format used by the archiver facility of Visual eForms Enterprise Server.

GetFormWindow

Description: Returns the handle of the form window in the ActiveX Control.

Syntax: C++ long CMmaFill::GetFormWindow()
Visual Basic [form.]MmaFill.GetFormWindow()

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

GetNextField

Description:	Returns the name of the next field in tabbing order on the form.
Syntax:	C++ VARIANT CMmaFill::GetNextField() Visual Basic [form.]MmaFill.GetNextField()
Parameter:	None
Remarks:	None
See Also:	GetFirstField()
Return Values:	Variant - Name of the next Field Null - An Error was encountered or no more fields were found on the form
Example:	See “GetFirstField” on page 234

GetNumPages

Description: Returns the total number of pages for the currently loaded form.

Syntax: C++ short CMmaFill::GetNumPages()
Visual Basic [form.]MmaFill.GetNumPages()

Parameters: None.

Remarks: None.

Return Values: > 0 - the Number of pages in the form
 0 - an error occurred

Example: See “GetFirstField” on page 234

GetSignatureTimestamp

Description: Returns a string holding Date and Time the Signature field was signed.

Syntax:

C++ VARIANT CMmaFill::GetSignatureTimestamp(LPCTSTR
 FieldName)

Visual Basic [form.]GetSignatureTimestamp(FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.

Remarks: None.

See Also: SignForm(), UnsignForm(), ValidateForm(), GetSignerName()

Return Values:

True - Success

False - An Error was encountered

GetSignerName

Description: Returns name of the person who signed into a Signature field (if available).

Syntax: C++ VARIANT CMmaFill::GetSignerName(LPCTSTR FieldName)
Visual Basic [form.]GetSignerName(FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.

Remarks: None.

See Also: SignForm(), UnsignForm(), ValidateForm(), GetSignatureTimestamp()

Return Values: True - Success
False - An Error was encountered

GetUnfilledMandatory

Description: Returns the name of the first field on the form that has been marked 'mandatory' and has not been filled with any value.

Syntax: C++ VARIANT CMmaFill::CMmaFill::GetUnfilledMandatory()
Visual Basic [form.]MmaFill.GetUnfilledMandatory()

Parameter: None

Remarks: none

Return Values: Variant - Name of the first unfilled Field on the form
 Null - An Error was encountered

Example Following example checks to see if there is a Mandatory field that is left blank. If so, then the user is prompted and focus is placed back on to the Mandatory field.

```
Sub SaveMyForm()  
Dim FieldName  
FieldName = mmafill.GetUnfilledMandatory ()  
If FieldName <> "" then  
    rv = mmafill1.GotoField (FieldName)  
    MsgBox "This is a Mandatory field and cannot be blank."  
End If  
End Sub
```

GetVersion

Description:	Returns the Version Number of Visual eForms Filler ActiveX.
Syntax:	<div>C++ CString CMmaFill:: GetVersion ()</div> <div>Visual Basic [form.]MmaFill. GetVersion ()</div>
Parameters:	None.
Remarks:	None.
Return Values:	String - Version of the Visual eForms ActiveX (e.g., "1,2,0,7")

GotoField

Description: Resets focus to field FieldName of currently loaded form.

Syntax: C++ `BOOL CMmaFill::GotoField(LPCTSTR FieldName)`
 Visual Basic `[form.]MmaFill.GotoField(Byval FieldName As String)`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.

Remarks: None.

Return Values: True - Call to this method was Successful
 False - An Error was encountered

Example: 'Copy the first 5 characters of LAST_NAME field and Paste it into
 'LAST_NAME_2 field

'Use DisableRedraw to eliminate flickering, if any

'Enable Redraw after the procedure

Dim rv

rv = MmaFill1.OpenFormDialog()

rv = MmaFill1.DisableRedraw(True)

rv = MmaFill1.GotoField("LAST_NAME")

rv = MmaFill1.SetCursorPosition(0, 5)

rv = MmaFill1.Copy()

rv = MmaFill1.GotoField("LAST_NAME_2")

rv = MmaFill1.Paste()

rv = MmaFill1.DisableRedraw(False)

GotoFieldByTabOrder

Description: Resets focus to the field with TabOrder FieldTaborder.

Syntax: C++ `BOOL CMmaFill::GotoFieldByTabOrder(long FieldTabOrder)`
Visual Basic `[form.]MmaFill.GotoFieldByTabOrder(Byval FieldTabOrder As Long)`

Parameter: The following parameters are available

Parameter	Description
FieldTabOrder	TabOrder of the desired Form Field

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

GotoPage

Description: Sets the current page of the form to PageNumber and sets the focus to the first field of the new page.

Syntax: C++ `BOOL CMmaFill::GotoPage(long PageNumber, BOOL NoScroll)`
Visual Basic `[form.]MmaFill.GotoPage(Byval PageNumberAs Long, NoScroll As Boolean)`

Parameter: The following parameters are available

Parameter	Description
PageNumber	Page Number to which we are switching to
NoScroll	Set to True to avoid resetting of the scrollbar to the top of the page; otherwise set to False.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

HighlightFields

Description: Disables or enables highlighting of fields on the currently loaded form.

Syntax: C++ `BOOL CMmaFill::HighlightFields(BOOL Enable)`
Visual Basic `[form.]MmaFill.HighlightFields(Byval Enable As Boolean)`

Parameter: The following parameters are available

Parameter	Description
Enable	set to True to highlight fields; otherwise set to False.

Remarks: This method is used to give the user a visual contrast between editable fields on the form and static text and graphic areas.

Once this method is called, the changes remain in effect for subsequent forms loaded into the control until the next call to this method.

Return Values: True - Call to this method was Successful
False - An Error was encountered

ImportAscii

Description: Opens a Form File in Ascii format and displays it in the ActiveX control.

Syntax: C++ `BOOL CMmaFill::ImportAscii(LPCTSTR FileName)`
Visual Basic `[form.]MmaFill.ImportAscii(Byval FileName As String)`

Parameter: The following parameters are available

Parameter	Description
FileName	Name of the Form File

Remarks: In addition to its binary format for Form Files (i.e., forms with “.far” extension), MMAFill ActiveX Control supports an Ascii format for object definition and properties. This method is particularly useful for converting forms from a proprietary electronic form environment to MMAFill’s “.far” format.

Return Values: True - Call to this method was Successful
False - An Error was encountered

IsFormChanged

Description:	Returns the internal Modified-Flag of the currently loaded form.
Syntax:	<div>C++ BOOL CMmaFill::IsFormChanged()</div> <div>Visual Basic [form.]MmaFill.IsFormChanged()</div>
Parameters:	None.
Remarks:	None.
Return Values:	<div>True - Form has been modified since it was loaded or AbandonChanges() was last called</div> <div>False - Form has NOT been modified since it was loaded or AbandonChanges() was last called</div>

IsFormLocked

Description:	Returns the internal Locked-Flag of the currently loaded form.
Syntax:	<div>C++ BOOL CMmaFill::IsFormLocked()</div> <div>Visual Basic [form.]MmaFill.IsFormLocked()</div>
Parameters:	None.
Remarks:	The internal Locked-Flag can be set via LockForm() Method. Note that these semantics associated with a Locked-Form are entirely up to the application developer.
Return Values:	<div>True - Form has been Locked</div> <div>False - Form has NOT been Locked</div>

LockForm

Description:	Sets the internal Locked-Flag of the currently loaded form.
Syntax:	<div>C++ BOOL CMmaFill::LockForm() Visual Basic [form.]MmaFill.LockForm()</div>
Parameters:	None.
Remarks:	The semantics associated with a Locked-Form are entirely up to the application developer.
Return Values:	<div>True - Call to this method was Successful False - An Error was encountered</div>

MAPISendMail

Description: Creates a New e-mail message and attaches the currently loaded form to it.

Syntax:

C++ `BOOL CMmaFill::MAPISendMail(LPCTSTR Subject, LPCTSTR Text, LPCTSTR Recipient, short Flags)`

Visual Basic `[form.]MAPISendMail(Subject As String, Text As String, Recipient As String, Flags As Integer) As Boolean`

Parameter: The following parameters are available

Parameter	Description
Subject	Subject portion of the email message
Text	Text Body of the email message
Recipient	email address of the recipient
Flags	Set to 0 to allow the user to edit the message; set to 1 to disallow editing.

Remarks: MAPISendMail is MAPI compliant (i.e., it ONLY works with MAPI compliant email programs such as OutLook).

See Also: None

Return Values:

True - Success

False - An Error was encountered

NextField

Description:	Advances the focus to the next field in the Tabbing Order of the currently loaded form.
Syntax:	C++ <code>BOOL CMmaFill::NextField()</code> Visual Basic <code>[form.]MmaFill.NextField()</code>
Parameters:	None.
Remarks:	None.
Return Values:	True - Call to this method was Successful False - An Error was encountered

OnPrintText

Description: Sets the Print Text and its corresponding attributes. Once set, the PrintText will be reflected on all form printouts. There is no effect on the screen view of the form.

Syntax:

```
C++      BOOL CMmaFill::OnPrintText(LPCTSTR Text, short x, short y,
                                         LPCTSTR FontName, short Height, short Escapement,
                                         BOOL Bold, BOOL Italic, BOOL Underline, long TextColor)

Visual Basic [form.]MmaFill.OnPrintText(Byval Text As String, Byval x As Integer, Byval y As Integer, Byval FontName As String,
                                         Byval Height As Integer, Byval Escapement As Integer,
                                         Byval Bold As Boolean, Byval Italic As Boolean,
                                         Byval Underline As Boolean, Byval TextColor As Long)
```

Parameter: The following parameters are available

Parameter	Description
Text	Print Text to be reflected on printouts
x	x Position of the text in twips (1440 twips = 1 inch)
y	y Position of the text in twips
FontName	Name of the Font to be used for Print Text
Height	Font Height in twips
Escapement	angle (in 0.1-degree units) between the escapement vector and the x-axis
Bold	Set to True for Bold Font
Italic	Set to True for Italic Font
Underline	Set to True for Underlined Font
TextColor	Color of Text as an RGB value

Remarks: This method is primarily used for adding watermarks to form printouts and it does not change the on-screen form. To disable, call the method with Text parameter set to an empty string (i.e., "").

Once this method is called, the changes remain in effect for subsequent forms loaded into the control until the next call to this method.

Call this method before printing.

You can call this method once. If called more than once, ONLY the last call takes effect.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example:

```
Dim rv
'Open a form
rv = MmaFill1.OpenFormDialog()
'Add a "DEMO" watermark with the following characteristics:
'X position = 4" (4x1440 twips)
'Y position = 4" (4x1440 twips)
'Font Type = Times New Roman or Arial
'Font Height = 1" (1x1440 twips)
'Angle: 45 Degrees = 45x0.1 increments
'Bold, Italic and Underline characters
'Color= Green - RGB value = (0,255,0)
'

rv = MmaFill1.OnPrintText("DEMO", 4 * 1440, 4 * 1440, "times new roman", 1
* 1440, 450, True, True, True, RGB(0, 255, 0))
rv = MmaFill1.PrintDialog()
End Sub
```

OpenForm

Description: Opens a Form File and displays it in the ActiveX control.

Syntax: C++ `BOOL CMmaFill::OpenForm(LPCTSTR FileName)`
Visual Basic `[form.]MmaFill.OpenForm(Byval FileName As String)`

Parameter: The following parameters are available

Parameter	Description
FileName	Full Pathname or URL address of the Form File

Remarks: This method can be used to open forms either from the local or network hard disks (e.g., "c:\myforms\form1.far" or "g:\formsdir\form2.far or \\nt_server\all forms\form3.far) in addition to forms residing on the intranet or the internet (e.g., <http://www.myfirm.com/forms/form4.far> or <ftp://ftp.mycomany.com/formsdir/form5.far>).

Return Values: True - Call to this method was Successful
False - An Error was encountered

OpenFormData

Description: Opens a Form Data File and displays it in the ActiveX control.

Syntax:

```
C++          BOOL CMmaFill::OpenFormData(LPCTSTR FileName,
                                           LPCTSTR FormsDir, LPCTSTR FormExt, BSTR FAR* Header)

Visual Basic [form.]MmaFill.OpenFormData(Byval FileName As String, Byval
                                           FormsDir As String, Byval FormExt As String, Header As String)
```

Parameter: The following parameters are available

Parameter	Description
FileName	Full Pathname or URL address of the Data File
FormsDir	Directory in which corresponding Form for the Data File resides
FormExt	Extension of the Form File excluding the period (Example: "far")
Header	Information previously saved into the Data File

Remarks: Used in conjunction with SaveFormData(), these two methods provide a simple mechanism for saving and retrieving data entered on a form. For simple applications, these two methods provide an excellent substitute for an elaborate database as the data repository.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

```
Dim rv
Dim header As String * 100, FormsDirectory As String
Dim FormsExtension As String
header = ""           'this is optional
FormsDirectory = ""   'this is optional
FormsExtension = ""   'this is optional
'
```

'User must first Open the Form (filename.far) before opening the data (c:\test.dat) file.

```
rv = MmaFill1.OpenFormData("c:\test.dat", FormsDirectory, FormsExtension,
header)
```

```
If Not rv Then
```

```
    MsgBox "Open Data Failed. Possible Problems are:" & Chr(10) &  
    Chr(13) & "1. Form is not open"
```

```
End If
```

OpenFormDataDialog

Description: Displays an Open Form Data Dialog allowing the user to select a Form Data File to be opened in the ActiveX Control.

Syntax: C++ `BOOL CMmaFill::OpenFormDataDialog(BSTR FAR* Header)`
Visual Basic `[form.]MmaFill.OpenFormDataDialog(Header As String)`

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered or the User pushed the Cancel button of the Dialog

Example:

```
Dim rv
Dim header As String * 100, FormsDirectory As String
Dim FormsExtension As String
header = ""           'this is optional
FormsDirectory = ""   'this is optional
FormsExtension = ""   'this is optional
'
'User must first Open the Form (filename.far) before opening the data
(filename.dat) file.
rv = MmaFill1.OpenFormDataDialog(header)
If Not rv Then
    MsgBox "Open Data Failed. Possible Problems are:" & Chr(10) &
    Chr(13) & "1. Form is not open"
End If
```

OpenFormDialog

Description:	Displays an Open Form Dialog allowing the user to select a Form File to be opened in the ActiveX Control.
Syntax:	C++ <code>BOOL CMmaFill::OpenFormDialog()</code> Visual Basic <code>[form.]MmaFill.OpenFormDialog()</code>
Parameters:	None.
Remarks:	None.
Return Values:	True - Call to this method was Successful False - An Error was encountered or the User pushed the Cancel button of the Dialog
Example:	'Display the "Open a Form" Dialog box so that user can select and open a form Dim rv rv = MmaFill1.OpenFormDialog()

Paste

Description: Inserts the contents of Windows clipboard into the current form field.

Syntax: C++ `BOOL CMmaFill::Paste()`
Visual Basic `[form.]MmaFill.Paste()`

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Copy the first 5 characters of LAST_NAME field and Paste it into
'LAST_NAME_2 field

'Use DisableRedraw to eliminate flickering, if any

'Enable Redraw after the procedure

Dim rv

rv = MmaFill1.OpenFormDialog()

rv = MmaFill1.DisableRedraw(True)

rv = MmaFill1.GotoField("LAST_NAME")

rv = MmaFill1.SetCursorPosition(0, 5)

rv = MmaFill1.Copy()

rv = MmaFill1.GotoField("LAST_NAME_2")

rv = MmaFill1.Paste()

rv = MmaFill1.DisableRedraw(False)

PrevField

Description:	Advances the focus to the previous field in the Tabbing Order of the currently loaded form.
Syntax:	C++ <code>BOOL CMmaFill::PrevField()</code> Visual Basic <code>[form.]MmaFill.PrevField()</code>
Parameters:	None.
Remarks:	None.
Return Values:	True - Call to this method was Successful False - An Error was encountered

Print

Description: Prints the currently loaded form with Addendum (if any) onto the supplied PrinterDevice Context.

Syntax:

```
C++          BOOL CMmaFill::Print(long printerDC, short StartPage, short
              EndPage, LPCTSTR Addendum)

Visual Basic [form.]MmaFill.Print(Byval printerDC As Long, Byval StartPage
              As Integer, Byval EndPage As Integer, Byval Addendum As
              String)
```

Parameter: The following parameters are available

Parameter	Description
printerDC	Printer Device Context
StartPage	Starting page to be printed
EndPage	Ending page to be printed
Addendum	Addendum Header to be printed on Addendum pages

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: “GetFieldAddendumText” on page 215

PrintAbort

Description:	Aborts the current Print Job.
Syntax:	C++ BOOL CMmaFill::PrintAbort() Visual Basic [form.]MmaFill.PrintAbort()
Parameters:	None.
Remarks:	This method is used in conjunction with PrintStart(), PrintEnd(), PrintPage()and PrintAddendum() methods to provide full control over the printing process.
Return Values:	True - Call to this method was Successful False - An Error was encountered
Example:	“GetFieldAddendumText” on page 215

PrintAddendum

Description: Prints the Addendum in the current Print Job.

Syntax: C++ `BOOL CMmaFill::PrintAddendum(LPCTSTR Addendum)`
Visual Basic `[form.]MmaFill.PrintAddendum(Byval Addendum As String)`

Parameter: The following parameters are available

Parameter	Description
Addendum	Addendum Header to be printed on Addendum pages

Remarks: This method is used in conjunction with PrintStart(), PrintEnd(), PrintPage() and PrintAbort() methods to provide full control over the printing process.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: "GetFieldAddendumText" on page 215

PrintDialog

Description:	Displays a Print Dialog allowing the user to first select a Printer Destination and then print the currently loaded form based on the Print Dialog settings.
Syntax:	C++ <code>BOOL CMmaFill::PrintDialog()</code> Visual Basic <code>[form.]MmaFill.PrintDialog()</code>
Parameters:	None.
Remarks:	None.
Return Values:	True - Call to this method was Successful False - An Error was encountered
Example:	<pre>Dim rv rv = MmaFill1.PrintDialog()</pre>

PrintEnd

Description:	Marks the end of the current Print Job.
Syntax:	<div>C++ Visual Basic</div> <div>BOOL CMmaFill::PrintEnd() [form.]MmaFill.PrintEnd()</div>
Parameters:	None.
Remarks:	This method is used in conjunction with PrintStart(), PrintAbort(),PrintPage() and PrintAddendum() methods to provide full control over the printing process.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
example:	“GetFieldAddendumText” on page 215

PrintForm

- Description:** Prints the currently loaded form without Addendum onto the supplied PrinterDevice Context.
- Syntax:**
- | | |
|--------------|--|
| C++ | BOOL CMmaFill::PrintForm(long printerDC, short StartPage, short EndPage) |
| Visual Basic | [form.]MmaFill.PrintForm(Byval printerDC As Long, Byval StartPage As Integer, Byval EndPageAs Integer) |
- Parameter:** The following parameters are available
- | Parameter | Description |
|-----------|-----------------------------|
| printerDC | Printer Device Context |
| StartPage | Starting page to be printed |
| EndPage | Ending page to be printed |
- Remarks:** This method is similar to Print() with the only difference that this method does not print an Addendum sheet.
- Return Values:**
- True - Call to this method was Successful
- False - An Error was encountered
- Example:** “GetFieldAddendumText” on page 215

PrintFreeDC

Description: Frees a printer DC (Device Context) previously created by PrintGetDC.

Syntax: C++ VOID CMmaFill::PrintFreeDC ()
Visual Basic [form.]MmaFill.PrintFreeDC ()

Parameter: None.

Remarks: This method is used in conjunction with PrintGetDC.

Return Values: None.

Example: See “GetFieldAddendumText” on page 215

PrintGetDC

Description: Returns a printer DC (Device Context).

Syntax: C++ LONG CMmaFill::PrintGetDC (LPCTSTR Options)
Visual Basic [form.]MmaFill.PrintGetDC (Byval Options as String) As Long

Parameter: The following parameters are available

Parameter	Description
Options	The Following parameter/value pairs are separated by a semicolon: "PrinterName=<UNC Printer-Path>" Specifies the printer. This should be in UNC format (e.g., \\SomeServer\Printer1). "Prompt=[0 or 1]" If 1, the user is prompted for a printer dialog. If 0, the user is not prompted and either the default or specified printer DC is returned. If not present, defaults to 1.

Remarks: This method is used in conjunction with PrintStart(), PrintForm() or Print() method to provide full control over the printing process.

Return Values:

- DC to the selected printer
- 0 if a DC could not be created

Example: See "GetFieldAddendumText" on page 215

PrintGetParams

Description: Returns users selection on the print dialog that is produced by PrintGetDC() function.

Syntax: C++ VARIANTE CMmaFill::PrintGetParams()
Visual Basic [form.]MmaFill.PrintGetParams() As String

Parameter: None.

Remarks: This method is used in conjunction with PrintGetDC(), PrintStart(), PrintAbort(), PrintEnd() and PrintAddendum() methods to provide full control over the printing process.

Return Values: An XML string that includes the page range and number of copies selected by the user.

Example: MmaFill.PrintGetDC ("")
MsgBox (MmaFill.PrintGetParams())

‘if the user selected pages 2 to 5, then this XML string is returned:

```
<?xml version="1.0"?><PRINTPARAMS><FROMPAGE>2</FROMPAGE><TOPAGE>5</TOPAGE><COPIES>1</COPIES></PRINTPARAMS>
```

‘if the user leave the default settings, then this XML string is returned:

```
<?xml version="1.0"?><PRINTPARAMS><FROMPAGE>-1</FROMPAGE><TOPAGE>-1</TOPAGE><COPIES>1</COPIES></PRINTPARAMS>
```

notice that both FROMPAGE and TOPAGE values are set to -1

PrintPage

Description: Prints one page to current Print Job.

Syntax: C++ `BOOL CMmaFill::PrintPage(short PageNum)`
Visual Basic `[form.]MmaFill.PrintPage(Byval PageNum As Integer)`

Parameter Description: The following parameters are available

Parameter	Description
PageNum	Page to be printed

Remarks: This method is used in conjunction with PrintStart(), PrintAbort(), PrintEnd()and PrintAddendum() methods to provide full control over the printing process.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: “GetFieldAddendumText” on page 215

Redraw

Description: Redraws (i.e., repaints) the current page of the currently loaded form.

Syntax: C++ void CMmaFill::Redraw()
Visual Basic [form.]MmaFill.Redraw()

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

SaveForm

Description: Packages the currently loaded form and data entered on the form into a new Form File.

Syntax: C++ `BOOL CMmaFill::SaveForm(LPCTSTR FileName)`
Visual Basic `[form.]MmaFill.SaveForm(Byval FileName As String)`

Parameter: The following parameters are available

Parameter	Description
FileName	Name of the new Form File

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

SaveFormData

Description: Saved the data on the currently loaded form into a Form Data File.

Syntax:

```
C++          BOOL CMmaFill::SaveFormData(LPCTSTR FileName,
                                           LPCTSTR Header)

Visual Basic [form.]MmaFill.SaveFormData(Byval FileName As String, Byval
                                           Header As String)
```

Parameter: The following parameters are available

Parameter	Description
FileName	Name of the Data File
Header	Header Information to be saved into the Data File

Remarks: Used in conjunction with OpenFormData(), these two methods provide a simple mechanism for saving and retrieving data entered on a form. For simple applications, these two methods provide an excellent substitute for an elaborate database as the data repository.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

```
Dim rv
Dim header As String * 100, FormsDirectory As String
Dim FormsExtension As String
```

```
header = ""           'this is optional
FormsDirectory = ""    'this is optional
FormsExtension = ""    'this is optional
```

'The following two methods could be used to save form's data to a DAT file.

```
rv = MmaFill1.SaveFormData("c:\test.dat", header)
```

```
If Not rv Then
```

```
    MsgBox "Save Data Failed."
```

```
End If
```

SaveFormDataDialog

Description: Displays a Save Form Dialog allowing the user to save the data on the currently loaded form into a Form Data File.

Syntax: C++ `BOOL CMmaFill::SaveFormDataDialog(LPCTSTR Header)`
 Visual Basic `[form.]MmaFill.SaveFormDataDialog (Byval Header As String)`

Parameter: The following parameters are available

Parameter	Description
Header	Header Information to be saved into the Data File

Remarks: Used in conjunction with `OpenFormDataDialog()`, these two methods provide a simple mechanism for saving and retrieving data entered on a form. For simple applications, these two methods provide an excellent substitute for an elaborate database as the data repository.

Return Values: True - Call to this method was Successful
 False - An Error was encountered or the User pushed the Cancel button of the Dialog

Example:

```
Dim rv
Dim header As String * 100, FormsDirectory As String
Dim FormsExtension As String
```

```
header = ""           'this is optional
FormsDirectory = ""   'this is optional
FormsExtension = ""   'this is optional
```

'The following two methods could be used to save form's data to a DAT file.

```
rv = MmaFill1.SaveFormDataDialog(header)
```

```
If Not rv Then
```

```
    MsgBox "Save Data Failed."
```

```
End If
```

SaveFormDialog

Description:	Displays a Save Form Dialog allowing the user to package the currently loaded form and data entered on the form into a new Form File.
Syntax:	C+ BOOL CMmaFill::SaveFormDialog() Visual Basic [form.]MmaFill.SaveFormDialog()
Parameter:	None
Remarks:	None.
Return Values:	True - Call to this method was Successful False - An Error was encountered or the User pushed the Cancel button of the Dialog
Example:	Dim rv rv = MmaFill1.SaveFormDialog()

Scroll

Description: Provides for manipulation of the Scroll Bar in the Filler window.

Syntax: C++ void Scroll(short ScrollType)
Visual Basic [form.]MmaFill.Scroll(ScrollType As Integer)

Parameters: Specifies the type of operation on the Scroll Bar according to the following table:

Value	Operation
1	Scroll vertically upward (small increment)
2	Scroll vertically downward (small increment)
3	Scroll vertically upward (large increment)
4	Scroll vertically downward (large increment)
5	Enable/show the vertical scroll bar
6	Disable/hide the vertical scroll bar
7	Scroll horizontally to the left (small increment)
8	Scroll horizontally to the right (small increment)
9	Scroll horizontally to the left (large increment)
10	Scroll horizontally to the right (large increment)
11	Enable/show the horizontal scroll bar
12	Disable/hide the horizontal scroll bar

Remarks: None.

Return Values: None.

Example: 'scroll the form down
Call MmaFill1.Scroll(2)

SetCursorPosition

Description: Positions the cursor or marks a block of text within the current field.

Syntax:

```
C++          BOOL CMmaFill::SetCursorPosition(short StartChar, short
              EndChar)

Visual Basic  [form.]MmaFill.SetCursorPosition(Byval StartChar As Integer,
              Byval EndChar As Integer)
```

Parameter: The following parameters are available

Parameter	Description
StartChar	Specifies the starting position (index is zero-based)
EndChar	Specifies the ending position

Remarks: If StartChar is 0 and EndChar is -1, all the text in the edit control is selected. If StartChar is -1, any current selection is removed.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

```
'Copy the first 5 characters of LAST_NAME field and Paste it into
'LAST_NAME_2 field

'Use DisableRedraw to eliminate flickering, if any
'Enable Redraw after the procedure
Dim rv
rv = MmaFill1.OpenFormDialog()
rv = MmaFill1.DisableRedraw(True)
rv = MmaFill1.GotoField("LAST_NAME")
rv = MmaFill1.SetCursorPosition(0, 5)
rv = MmaFill1.Copy()
rv = MmaFill1.GotoField("LAST_NAME_2")
rv = MmaFill1.Paste()
rv = MmaFill1.DisableRedraw(False)
```

SetFieldData

Description: Sets the value of field FieldName to FieldData.

Syntax:

C++ `BOOL CMmaFill::SetFieldData(LPCTSTR FieldName, LPCTSTR FieldData)`

Visual Basic `[form.]MmaFill.SetFieldData(Byval FieldName As String, Byval FieldData As String)`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the Form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName parameter of this method.
FieldData	New Data for field FieldName

Remarks: None.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

Example:

‘Put “523-09-2134” into the SSN field

Dim rv

rv = MmaFill1.SetFieldData(“SSN”, “523-09-2134”)

SetEnterpriseParams

Description: RESERVED FOR FUTURE.

Syntax:

Parameter:

Remarks:

Return Values:

Example:

SetFieldDataEx

Description: Sets the value of field FieldName to FieldData.

Syntax:

C++ `BOOL CMmaFill:: SetFieldDataEx (LPCTSTR FieldName,
 LPCTSTR FieldData, BOOL bRecalcDependants)`

Visual Basic `[form.]MmaFill. SetFieldDataEx (FieldName As String,
 FieldData As String, bRecalcDependants As Boolean)`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field
FieldData	New Data for field FieldName
BRecalcDependants	True: to allow for fields dependant on this Field's value to be recalculated after the call to SetFieldDataEx(); False: otherwise.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Put "523-09-2134" into the SSN field and Recalculate all dependencies

 Dim rv

 rv = MmaFill1.SetFieldData("SSN", "523-09-2134",TRUE)

SetFieldProperty

Description: Assigns PropertyVal to the field property PropertyID.

Syntax:

```
C++      BOOL CMmaFill::SetFieldProperty(LPCTSTR FieldName, long
        PropertyID, LPCTSTR PropertyVal)

Visual Basic [form.]MmaFill.SetFieldProperty(Byval FieldName As String,
        Byval PropertyID As Long, Byval PropertyVal As String)
```

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field. "*" applies the method to ALL fields of the form
PropertyID	ID of the property whose value is being reset (See below for valid IDs)
PropertyVal	New Property Value being assigned to Property

Following is a list of all available Property IDs and their values:

Action	ID	Value
Back Color	8	RGB of the Color (e.g., "255,0,0" for Red)
Line Color	9	RGB of the Color (e.g., "0,255,0" for Green)
Left Border	10	"1" Left Border ON "0" Left Border OFF
Top Border	11	"1" Top Border ON "0" Top Border OFF
Right Border	12	"1" Right Border ON "0" Right Border O
Bottom Border	13	"1" Bottom Border ON "0" Bottom Border O
Rounded Border	14	"1" Rounded Borders ON "0" Rounded Borders OFF
Text Color	26	RGB of the Color (e.g., "0,0,255" for Blue)
Overwrite	28	"1" Overwrite property ON "0" Overwrite property OFF

Action	ID	Value
Visible	76	"1" if the Field is Visible "0" otherwise
NotifyClick	29	"1" turn NotifyClick property ON "0" otherwise
NotifyModify	30	"1" turn NotifyModify property ON "0" otherwise
NotifyDbClick	31	"1" turn NotifyDbClick property ON "0" otherwise
GotFocus	32	"1" turn GotFocus property ON "0" otherwise
LostFocus	33	"1" turn LostFocus property ON "0" otherwise
MouseEnter	34	"1" turn MouseEnter property ON "0" otherwise
MouseExit	35	"1" turn MouseExit property ON "0" otherwise
MaxFillChars	27	Set/Reset the MaxFillChars property of the field

Remarks: 'SetFieldProperty' can be used to set all fields on a form with the specified new property value by using '*' for the field name parameter.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Set Background Color of the field "SSN" to RED
'

Dim rv
rv = MmaFill1.SetFieldProperty ("SSN",8,"255,0,0")

SetFocus

Description: Sets the focus of the Windows environment to this ActiveX Control.

Syntax: C++ void CMmaFill::SetFocus()
Visual Basic [form.]MmaFill.SetFocus()

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: ‘Set focus to the form.
 Call MmaFill1.SetFocus()

SetFormProperty**Description:**

Assigns PropertyValue to the form property PropertyID.

Syntax:

C++ `BOOL CMmaFill::SetFormProperty(long PropertyID,
 LPCTSTR PropertyValue)`

Visual Basic `[form.]MmaFill.SetFormProperty(Byval PropertyID As Long,
 Byval PropertyValue As String)`

Parameter:

The following parameters are available

Parameter	Description
PropertyID	ID of the property whose value is being assigned (See below for valid IDs)
PropertyValue	new Property Value being assigned to Property

PropertyID:

The following Property IDs are available

ID	Action	Value
106	Set Active-Field Color	RGB of the Color (e.g., "255,0,0" for Red)
107	Set Background Color	RGB of the Color (e.g., "0,255,0" for Green)
108	Set Form Orientation	"1" set the Form Orientation to Portrait "0" set the Form Orientation to Landscape
109	Show	"1" display Progress Window
	TCPIP Progress-Window	"0" do not display Progress Window
115	Allow Content Search	"1"
116	Track History	"1"
103	Set Form Width	Width of the form in inches.
104	Set Form Height	Height of the form in inches.
100	Set Form Name	Name property of the form.
101	Set Form Description	Description property of the form
102	Set Form Version	Version of the form used by version control facility of Visual eForms Enterprise Server.
112	Set Author	Set to the name of the person who designed the form.

ID	Action	Value
113	Set Category	Assign a category to the form.
114	Set Search Keywords	Comma-delimited list of keywords used by search facility of Visual eForms Enterprise Server.
117	Set Copyright	Copyright clause to be added to the form.
118	Set Comments	Comments of the designer.
119	Index Fields	Comma-delimited list of fields used by the archiver facility of Visual eForms Enterprise Server.
120	Set the Archive Format	File format used by the archiver facility of Visual eForms Enterprise Server.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Set Active-Field color to RED
Dim rv.
rv = MmaFill1.SetFormProperty (106, "255,0,0")

SetNotifyOnCalc

Description: Modifies the default behavior of the ActiveX Control on performing form-level calculations. If Notify is set to True, instead of doing the calculations at the form-level, only a NotificationEvent is sent to the holder of the Control.

Syntax: C++ `BOOL CMmaFill::SetNotifyOnCalc(BOOL Notify)`
Visual Basic `[form.]MmaFill.SetNotifyOnCalc(Byval Notify As Boolean)`

Parameter: The following parameters are available

Parameter	Description
Notify	set to True to disable performing form-level calculations

Remarks: This method is primarily used to perform the form-level calculations in the application code instead. In order to restore the default behavior of the ActiveX Control (i.e., performing form-level calculations automatically), call this method with Notify set to False.

Return Values: True - Call to this method was Successful
False - An Error was encountered

SetSharedFontTable

Description: Sets the Shared Font Table to provided FontFile.

Syntax: C++ `BOOL CMmaFill::SetSharedFontTable(LPCTSTR FontFile)`
Visual Basic `[form.]MmaFill.SetSharedFontTable(Byval FontFile As String)`

Parameter: The following parameters are available

Parameter	Description
FontFile	Name of the Font File

Remarks: This method is primarily used for forms whose Font Table is separate from the Form File and is shared between a set of forms.

Return Values: True - Call to this method was Successful
False - An Error was encountered

ShowNonPrintables

Description: Shows or hides the non-printable objects on the currently loaded form.

Syntax: C++ `BOOL CMmaFill::ShowNonPrintables(BOOL Show)`
Visual Basic `[form.]MmaFill.ShowNonPrintables(Byval Show As Boolean)`

Parameter: The following parameters are available

Parameter	Description
Show	set to True to show non-printable object on the form and False otherwise

Remarks: ShowNonPrintables() affects the Visible properties of the object. For example, ShowNonPrintables(FALSE) will turn the Visible property of the object to FALSE, making the object invisible.

Return Values: True - Call to this method was Successful
False - An Error was encountered

Example: 'Hide Non-Printable objects
Dim rv.
rv = MmaFill1.ShowNonPrintables (FALSE)

SignForm

Description: Signs the currently loaded form in Signature Field 'FieldName'.

Syntax:

```
C++          BOOL SignForm(LPCTSTR FieldName, LPCTSTR DisplayVal,
                           short Flags)

Visual Basic [form.]MmaFill.SignForm(FieldName As String, DisplayVal As
                           String, Flags AsInteger)
```

Parameter Description: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.
DisplayVal	Reserved for future use
Flags	Reserved for future use

Remarks: FieldName can have one of the following 5 types:

- NT Domain
- Entrust
- Hand Signature
- PKI
- User Defined

If signature field 'FieldName' is of type 'NT Domain', 'Entrust', or 'PKI', after a successful call to this method, the full name of the signer will be displayed in 'FieldName'. Turn ON the DblClick Notify flag of the signature field and then call the SignForm method on this event.

If type is 'Hand Signature', SignForm method is automatically called as soon as the Signature field loses focus and the signature drawn by the user into 'FieldName' will be finalized.

If type is 'NT Domain', FieldName will get the value from login user/account name in the NT Domain.

If type is 'Hand Signature', user can write into (sign) FieldName using mouse or mouse-pen.

If type is 'PKI', the user's signature will be taken from 'My Store' DigitalCertificate repository on the system.

Signature Type of 'User Defined' is reserved for future use.

Return Values:

True - Success

False - An Error was encountered

See Also:

UnSignForm(), ValidateForm(), GetSignerName(), GetSignatureTimestamp()

Example:

‘Add a signature field to your form and name it “Signature1”

‘turn the “DbClick” Notify property of “Signature1” to ON

‘When user DbClicks into “Signature1” I want to do one of 2 operations:

‘1. If “Signature1” is empty, then I want to Sign

‘2. If “Signature1” is NOT empty (it is already signed), I want to Unsign

‘When user DbClicks onto the “Signature1” field, the “FieldDbClick” event is

‘called. Within the “FieldDbClick” event we will perform our operation.

‘Note that “FieldDbClick” event returns the name of the field that user

‘DbClicks on (“Signature1”)

```
Private Sub MMAFill1_FieldDbClick(ByVal FieldName As String)
```

```
Dim rv
```

```
‘Read the data in FieldName. If empty, Sign; else, Unsign
```

```
If VarGetFieldString (FieldName) <> "" then
```

```
    rv = SignForm (FieldName, "", 0)
```

```
Else
```

```
    rv = UnsignForm(FieldName)
```

```
End If
```

```
End Sub
```

Undo

Description: Reverses the last edit in the current field.

Syntax: C++ `BOOL CMmaFill::Undo()`
Visual Basic `[form.]MmaFill.Undo()`

Parameters: None.

Remarks: None.

Return Values: True - Call to this method was Successful
False - An Error was encountered

UnsignForm

Description: Unsigns an already signed Signature field.

Syntax: C++ `BOOL CMmaFill::UnsignForm(LPCTSTR FieldName)`
 Visual Basic `[form.]UnsignForm(FieldName As String) As Boolean`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Parameter of this method.

Remarks: If “Lock Fields” property of the Signature field is set to TRUE in the Designer, UnsignForm() will unlock all fields associated with the Signature fields.

See Also: SignForm(), ValidateForm(), GetSignerName(), GetSignatureTimestamp()

Return Values: True - Success
 False - An Error was encountered

Example: ‘Add a signature field to your form and name it “Signature1”
 ‘turn the “DbClick” Notify property of “Signature1” to ON
 ‘When user DbClicks into “Signature1” I want to do one of 2 operations:
 ‘1. If “Signature1” is empty, then I want to Sign
 ‘2. If “Signature1” is NOT empty (i.e., it is already signed), I want to Unsign
 ‘When user DbClicks onto the “Signature1” field, the “FieldDbClick” event is
 ‘called. Within the “FieldDbClick” event we will perform our operation.
 ‘Note that “FieldDbClick” event returns the name of the field that user
 ‘DbClicks on (i.e., “Signature1”)
 Private Sub MmaFill1_FieldDbClick(ByVal FieldName As String)
 Dim rv
 ‘Read the data in FieldName. If empty, Sign; else, Unsign
 If VarGetFieldString (FieldName) <> “” then
 rv = SignForm (FieldName, “”, 0)
 Else

```
        rv = UnsignForm(FieldName)
    End If
End Sub
```

ValidateSignature

Description: Validates the Digital Signature in the Signature field against the form and the field data associated with that Signature field.

Syntax: C++ `BOOL CMmaFill::ValidateSignature(LPCTSTR FieldName)`
Visual Basic `[form.]ValidateSignature(FieldName As String) As Boolean`

Parameter: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Parameter of this method.

Remarks: If the Signature does not validate, in addition to setting the return value to False, an error message may also be displayed.

See Also: SignForm(), UnsignForm(), GetSignerName(), GetSignatureTimestamp()

Return Values: True - Signature Validated
False - Signature Did Not Validate

Example: ‘Add a signature field to your form and name it “Signature1”
‘Call this method to Validate “Signature1”

```
Dim rv
If ValidateSignature(“Signature1”) = TRUE then
    MsgBox “Signature is Validated.”
Else
    MsgBox “Signature is NOT Validated.”
End If
```


VarGetCurrField

Description:	Returns the name of the current field in focus.
Syntax:	C++ VARIANT CMmaFill::VarGetCurrField() Visual Basic [form.]MmaFill.VarGetCurrField()
Parameter:	None
Remarks:	none
See Also:	“GetCurrField” on page 212
Return Values:	Variant - Name of the Current Field Null - An Error was encountered
Example:	'Two ways of capturing the name of field that has Focus: 'GetCurrField() and VarGetCurrField() Dim rv Dim FieldName As String * 255 rv = MmaFill1.OpenFormDialog() rv = MmaFill1.GetCurrField(FieldName) MsgBox RTrim(FieldName) + " has now focus." - MsgBox MmaFill1.VarGetCurrField() + " has now focus."

VarGetFieldAddendumText

Description: Returns Addendum Text of field FieldName.

Syntax:

C++ VARIANT CMmaFill::VarGetFieldAddendumText(LPCTSTR
 FieldName)

Visual Basic [form.]MmaFill.VarGetFieldAddendumText (Byval FieldName
 As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks: Addendum text of a field refers to excess text that cannot be fit within the bounds of that field.

See Also: “GetFieldAddendumText” on page 215

Return Values: Variant - Addendum Text of the Field FieldName
Null - An Error was encountered

VarGetFieldHelp

Description: Returns Help Text of field FieldName.

Syntax: C++ VARIANT CMmaFill::VarGetFieldHelp(LPCTSTR FieldName)
Visual Basic [form.]MmaFill.VarGetFieldHelp (Byval FieldName As String)

Parameter Description: The following parameters are available

Parameter	Description
FieldName	Name or TabOrder of the DropList Field on the form. For TabOrder, prefix the number representing the Tab Order with “@”; for example, to address a field at Tab Order 12, pass “@12” as FieldName Paramater of this method.

Remarks: Help Text for form fields are set at Design time by assigning appropriate text to FieldHelp property of form fields.

See Also: “GetFieldHelp” on page 220

Return Values: Variant - Help Text for field FieldName
Null - An Error was encountered

VarGetFieldString

Description: Returns the value of field FieldName.

Syntax: C++ VARIANT CMmaFill::VarGetFieldString(LPCTSTR FieldName)
Visual Basic [form.]MmaFill. VarGetFieldString (Byval FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks: None.

See Also: “GetFieldString” on page 232

Return Values: Variant - Value of field FieldName
Null - An Error was encountered

VarOpenFormData

Description: Opens a Form Data File and displays it in the ActiveX control.

Syntax:

```
C++          VARIANT CMmaFill::OpenFormData(LPCTSTR FileName,
          LPCTSTR FormsDir, LPCTSTR FormExt)

Visual Basic [form.]MmaFill.OpenFormData(Byval FileName As String, Byval
          FormsDir As String, Byval FormExt As String) As String
```

Parameter: The following parameters are available

Parameter	Description
FileName	Full Pathname or URL address of the Data File
FormsDir	Directory in which corresponding Form for the Data File resides
FormExt	Extension of the Form File excluding the period (Example: "far")

Remarks: Used in conjunction with SaveFormData(), these two methods provide a simple mechanism for saving and retrieving data entered on a form. For simple applications, these two methods provide an excellent substitute for an elaborate database as the data repository.

Return Values: String - The Header information stored in the Form Data File
Null - Failure occurred

See Also: "OpenFormData" on page 261

Example:

```
Dim rv
Dim FormsDirectory As String
Dim FormsExtension As String
FormsDirectory = ""      'this is optional
FormsExtension = ""     'this is optional
'

'User must first Open the Form (filename.far) before opening the data
(c:\test.dat) file.

rv = MmaFill1.VarOpenFormData("c:\test.dat", FormsDirectory, FormsEx-
tension)
```

```
If rv = "" Then
    MsgBox "Open Data Failed. Possible Problems are:" & Chr(10) &
    Chr(13) & "1. Form is not open"
End If
```

VarOpenFormDataDialog

Description:	Displays a dialog allowing the user to select a Form Data File to be opened in the ActiveX Control.
Syntax:	<div>C++ CString CMmaFill:: VarOpenFormDataDialog ()</div> <div>Visual Basic [form.]MmaFill.VarOpenFormDataDialog ()</div>
Parameters:	None.
Remarks:	None.
See Also:	“OpenFormDataDialog” on page 263
Return Values:	String - The Header information stored in the Form Data File.

ViewEnlarge

Description:	Changes the Zoom level of the current form to enlarged mode, which is approximately double the actual form size.
Syntax:	C++ <code>BOOL CMmaFill::ViewEnlarge()</code> Visual Basic <code>[form.]MmaFill.ViewEnlarge()</code>
Parameters:	None.
Remarks:	None.
See Also:	<code>ViewFitSize()</code> , <code>ViewRealSize()</code> , <code>ZoomFactor</code>
Return Values:	True - Call to this method was Successful False - An Error was encountered

ViewFitSides

Description:	Changes the Zoom level of the current form so that the form is displayed with both sides flush against the form window sides.
Syntax:	C++ <code>BOOL CMmaFill::ViewFitSides()</code> Visual Basic <code>[form.]MmaFill.ViewFitSides()</code>
Parameters:	None.
Remarks:	None.
See Also:	<code>ViewRealSize()</code> , <code>ViewEnlarge()</code> , <code>ZoomFactor</code>
Return Values:	True - Call to this method was Successful False - An Error was encountered

ViewRealSize

Description:	Changes the Zoom level of the current form so that the form to the actual form size.
Syntax:	<div>C++ Visual Basic</div> <div>BOOL CMmaFill::ViewRealSize() [form.]MmaFill.ViewRealSize()</div>
Parameters:	None.
Remarks:	None.
See Also:	ViewFitSize(), ViewEnlarge(), ZoomFactor
Return Values:	True - Call to this method was Successful False - An Error was encountered

XMLGetFormData

- Description:** Returns an XML-encoded data stream, in the form of a string, containing the Form Data.
- The XML-encoded data stream, if password-encrypted, is compliant with the current W3 org draft standard for XML encryption.
- Syntax:**
- C++ VARIANT CMmaFill::XMLGetFormData (LPCTSTR Options)
- Visual Basic [form.]MmaFill.XMLGetFormData (Options As String)
- Parameters:** The following parameters are available

Parameter	Description
Options	<p>The following parameter/value pairs are separated by semi colon:</p> <ul style="list-style-type: none">• EnableExtAttr=[0 1] Setting EnableExtAttr to 1 will allow modified properties of the fields to also be included in the returned XML string. The default value of this parameter is 1.• Encrypt =<enable> enables encryption of the data (default is false) For Example: Encrypt=1• EncryptAlg =<encryption algorithm type> encryption algorithm. Choices are 3DES or RC4 (default is 3DES) For Example: EncryptAlg=3DES• EncryptKeyLen =<No. of Bits> number of bits for the encryption key (note some algs such as 3DES ignore this parameter) (default is 128) For Example: EncryptKeyLen=128• EncryptPwd =<password> password used to encrypt the XML data with

Parameters:

(Parameters continued)

Parameter	Description
Options	<p>The following parameter/value pairs are separated by semi colon:</p> <ul style="list-style-type: none"> • Pages=<Page Number> <Page Number> is a single page number If "Pages" parameter is omitted, by default all pages will be included in the returned XML string. • SkipBlanks=[0 or 1] 0=Don't Skip, 1=Skip If "SkipBlanks" parameter is omitted, by default only fields without empty value will be included in the return XML string.

Remarks:

The DTD (Document Type Definition) is as follows:

```

<!ELEMENT FORMDATA
(VERSION,FORMNAME,FORMLOC,FORMVERSION,HEADER,ENCRYPTIO
N,FIELDDATA) >
<!ELEMENT VERSION (#PCDATA) >
<!ELEMENT FORMNAME (#PCDATA)>
<!ELEMENT FORMLOC (#PCDATA)>
<!ELEMENT FORMVERSION (#PCDATA)>
<!ELEMENT HEADER (#PCDATA)>
<!ELEMENT ENCRYPTION (#PCDATA)>
<!ELEMENT FIELDDATA (F+)>
<!ELEMENT F (#PCDATA)>
<!ATTLIST F NAME CDATA #REQUIRED>

```

Return Values:

String - XML-encoded stream of data

Example:

```

XMLString = XMLGetFormData( " " )
XMLString = XMLGetFormData( "Pages=1;SkipBlanks=1" )
XMLString = XMLGetFormData( "pages=2" )
XMLString =
XMLGetFormData( "Pages=1;SkipBlanks=1;Encrypt=1;EncryptAlg=3DES;
EncryptKeyLen=128;EncryptPwd=myspassword" )

```


Parameter	Description
Options	<ul style="list-style-type: none">• "EncryptPwd =<password>" EncryptPwd the password to decrypt the data with• "ClearData=[0 or 1]" 0 =Do NotClear Data,1 =ClearData If "ClearData" parameter is omitted, by default all form fields are cleared prior to processing the XML String• "FormLoad=[0 or 1]" 0 =Do NotLoadForm,1 =Load Form If "FormLoad" parameter is omitted, by default the form referenced in the XML string will be loaded prior to processing the XML string if the currently loaded form in the Filler ActiveX is different from the one referenced by XML String• "VersionCheckWarning=[0 or 1]" If 0, a Version-Check Warning will not be displayed in case the XML data being passed as a parameter to this method belongs to an older version of the currently loaded form. If not present, defaults to 0.

Remarks: See "XMLGetFormData" on page 316 for a DTD (Document Type Definition).

Return Values: True - Call to this method was Successful
False - An Error was encountered

Examples: XMLSetFormData(XMLString, "")
XMLSetFormData(XMLString, "Formload=1;ClearData=0")
XMLSetFormData(XMLString, "ClearData=1")

```
XMLSetFormData("XMLString, " Formload =1; EncryptPwd =mypassword")
```

Events

FieldClick

Description: Occurs when the user presses and then releases a mouse button over a form field.

Syntax:

C++	<code>afx_msg void OnFieldClickMmaFill(LPCTSTR FieldName)</code>
Visual Basic	<code>Sub MmaFill_FieldClick(Byval FieldName As String)</code>

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks: You must set the Notify | Click property of the desired Form Field to True for the FieldClick event to occur for that Form Field.

FieldDbClick

Description: Occurs when the user presses and then releases a mouse button and then presses and releases it again over a form field.

Syntax:

C++	afx_msg void OnFieldDbClickMmaFill(LPCTSTR FieldName)
Visual Basic	Sub MmaFill_FieldDbClick(Byval FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks: You must set the Notify | DbClick property of the desired Form Field to True for the FieldDbClick event to occur for that Form Field.

FieldModified**Description:**

Occurs when the user leaves a form field that has been modified.

Syntax:

C++ afx_msg void OnFieldModifiedMmaFill(LPCTSTR FieldName)

Visual Basic Sub MmaFill_FieldModified(Byval FieldName As String)

Parameter:

The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks:

You must set the Notify | Modify property of the desired Form Field to True for the FieldModified event to occur for that Form Field.

FieldGotFocus**Description:**

Occurs when the user sets the focus to a form field.

Syntax:

C++ afx_msg void OnFieldGotFocusMmaFill(LPCTSTR FieldName)

Visual Basic Sub MmaFill_FieldGotFocus(Byval FieldName As String)

Parameter:

The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks:

You must set the Notify | GotFocus property of the desired Form Field to True for the FieldGotFocus event to occur for that Form Field.

FieldLostFocus

Description: Occurs when the user leaves a form field.

Syntax:

C++	afx_msg void OnFieldLostFocusMmaFill(LPCTSTR FieldName)
Visual Basic	Sub MmaFill_FieldLostFocus(Byval FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks: You must set the Notify | LostFocus property of the desired Form Field to True for the FieldLostFocus event to occur

FieldMouseEnter

Description: Occurs when the user moves the mouse over and into a form field.

Syntax:

```
C++          afx_msg void OnFieldMouseEnterMmaFill(LPCTSTR  
              FileName)  
Visual Basic  Sub MmaFill_FieldMouseEnter(Byval FileName As String)
```

Parameter: The following parameters are available

Parameter	Description
FileName	Name of the Form Field

Remarks: You must set the Notify | MouseEnter property of the desired Form Field to True for the FieldMouseEnter event to occur for that Form Field.

FieldMouseExit

Description: Occurs when the user moves the mouse out of a form field.

Syntax:

C++ afx_msg void OnFieldMouseExitMmaFill(LPCTSTR
 FieldName)

Visual Basic Sub MmaFill_FieldMouseExit(Byval FieldName As String)

Parameter: The following parameters are available

Parameter	Description
FieldName	Name of the Form Field

Remarks: You must set the Notify | MouseExit property of the desired Form Field to True for the FieldMouseExit event to occur for that Form Field.

FillerLoaded

Description:	Occurs when the filler ActiveX is instantiated.
Syntax:	<div>C++ afx_msg void OnFillerLoadedMmaFill()</div> <div>Visual Basic Sub MmaFill_FillerLoaded()</div>
Remarks:	FillerLoaded() event occurs when the ActiveX is fully loaded and instantiated.

GotFocus**Description:**

Occurs when the filler ActiveX gets the focus.

Syntax:

C++ afx_msg void OnGotFocusMmaFill()

Visual Basic Sub MmaFill_GotFocus()

Remarks:

As user switches between applications, the ActiveX can lose and regain focus. This event occurs as soon as the ActiveX regains the focus.

LostFocus

Description: Occurs when the filler ActiveX loses its focus to another application or object.

Syntax:

C++	<code>afx_msg void OnLostFocusMmaFill()</code>
Visual Basic	<code>Sub MmaFill_LostFocus()</code>

Remarks: This event occurs as user switches from the eForms ActiveX to another object or application.

OnChar

Description: Occurs when the user presses and releases a key or key combination.

Syntax:

```
C++     	afx_msg void OnChar(long FAR* nChar, short FAR* nRepCnt,
          	long FAR* nFlags, BOOL FAR* bCancel)

Visual Basic  Sub MmaFill_OnChar(nChar As Long, nRepCnt As Integer,
          	nFlags As Long, bCancelAs Boolean)
```

Parameter: The following parameters are available

Parameter	Description
nChar	Contains the character code value of the key.
NRepCnt	Contains the repeat count, the number of times the keystroke is repeated when user holds down the key.
nFlags	Contains the scan code, key-transition code, previous key state, and context code, as shown in the following Table.
bCancel	TRUE - Inhibit the character to be passed on to filler. FALSE - Default.

Table: Description of nFlags bits 0-15

nFlag Bits	Bits Description
0 - 7	Scan code (OEM-dependent value)
8	Extended key, such as a function key or a key on the numeric keypad (1 if it is an extended key; otherwise 0)
9 - 10	Not used
11 - 12	Used internally by Windows
13	Context code (1 if the ALT key is held down while the key is pressed; otherwise 0)
14	Previous key state (1 if the key is down before the call; 0 if the key is up)
15	Transition state (1 if the key is being released; 0 if the key is being pressed)

Remarks:

Although the OnChar event occurs when most keys are pressed, they are typically used to recognize or distinguish between:

- Extended character keys, such as function keys.
- Navigation keys, such as HOME, END, PAGE UP, PAGE DOWN, UP ARROW, DOWN ARROW, RIGHT ARROW, LEFT ARROW, and TAB.
- Combinations of keys and standard keyboard modifiers (SHIFT, CTRL, or ALT keys).
- The numeric keypad and keyboard number keys.

OnError

Description This event will be fired for all asynchronous error situations such as entering incorrect date format into a date field. This new feature enables programmers to fully customize the behavior of their applications in error situations.

Syntax

C++	<code>afx_msg void OnErrorMmaFill(long FAR* ErrorCode, LPCTSTR ErrorDesc)</code>
Visual Basic	<code>Sub MmaFill1_OnError(ByVal ErrorCode As Long, ByVal ErrorDesc As String)</code>

Parameters The following parameters are available:

Parameter	Description
ErrorCode	Numeric value of the Error
ErrorDesc	Textual description of the Error

Remarks None.

PageChange

Description: Occurs when the user moves to a different page.

Syntax:

C++	afx_msg void OnPageChangeMmaFill(short PageNum)
Visual Basic	Sub MmaFill_PageChange(Byval PageNum As Integer)

Parameter: The following parameters are available

Parameter	Description
PageNum	Page number to which the user has moved to

Remarks: None.

Database ActiveX

Properties

Caption Property

Description:	Gets or sets the CAPTION associated with the ActiveX control.
Syntax:	<div>C++<pre>CString CMmaADOi::Caption () void CMmaADOi::Caption (LPCTSTR caption)</pre></div> <div>Visual Basic<pre>[form.]MmaADOi.Caption</pre></div>
Remarks:	By default, Caption is set to the name of the object.
Data Type:	String

Methods

AboutBox

Description:	Displays the About box for the control.
Syntax:	<div>C++ VOID CMmaADOi::AboutBox();</div> <div>Visual Basic [form.] MmaADOi.AboutBox()</div>
Parameters:	None
Remarks:	This is the same as clicking About in the Properties window.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
See Also:	“GetLastError” on page 344

AddNew

Description:	Adds a New Record to the database.
Syntax:	<div>C++ Visual Basic</div> <div>BOOL CMmaADOi::AddNew(); [form.] MmaADOi.AddNew()</div>
Parameters:	None
Remarks:	Use this method if you do not want to use the ActiveX's own User Interface for adding a record to the database. Note that modifications to the form are not committed to the database until the Update method of the ActiveX is invoked.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
See Also:	“GetLastError” on page 344

Connect

Description: Establishes a connection between a form and its designated Database.

Syntax:

```
C++          BOOL CMmaADOi::Connect(LPCTSTR FormPath, LPDIS  
              PATCH FillerControl, LPCTSTR ConnectStr);
```

Visual Basic [form.] MmaADOi.Connect(FormPath As String, FillerControl
As Object, ConnectStr As String)

Parameters: The following parameters are available

Parameter	Description
FormPath	Full path of the form (.far file) to which Database connectivity is to be established.
FillerControl	Filler ActiveX control in which the form referred to by 'FormPath' will be loaded.
ConnectStr	Used to override the Database Relations parameters set in the Designer. To override database for all of the Data Sources assigned to a form, use the following syntax: "Database=<New Value>". For example: "Database=c:\mydatabases\clients.mdb To override individual Data Sources, prefix the Database keyword with "PrimaryDS" for the Primary Data Source and "SecondaryDS<#>" for the corresponding Secondary Data Source. For example: "PrimaryDS.Database=c:\DBs\db1.mdb;SecondaryDS1.Database=c:\DBs\db2.mdb"

Remarks: Use this method at application startup time to initiate loading of a form and establishing a connection to its designated Database. Note that Connect will fail if the Database Relations file (i.e., the file with ".drf" extension which is created by "Database Relations" facility of the Designer) is not in the same directory as the form file (i.e., ".far" file).

Return Values:

True - Call to this method was Successful

False - An Error was encountered

See Also: GetLastError(), Disconnect()

CreateDatabase

Description:: Create a new instance of the database associated with the form 'FormPath'.

Syntax:

```
C++          BOOL CMmaADOi::CreateDatabase(LPCTSTR FormPath,
                                           LPDISPATCH FillerControl, LPCTSTR ConnectStr);
```

Visual Basic [form.] MmaADOi.CreateDatabase (FormPath As String, Filler
Control As Object, ConnectStr As String)

Parameters: The following parameters are available

Parameter	Description
FormPath	Full path of the form (.far file) to which Database connectivity is to be established.
FillerControl	Filler ActiveX control in which the form referred to by 'FormPath' will be loaded.
ConnectStr	Used to override the Database Relations parameters set in the Designer. To override database for all of the Data Sources assigned to a form, use the following syntax: "Database=<New Value>". For example: "Database=c:\mydatabases\clients.mdb To override individual Data Sources, prefix the Database keyword with "PrimaryDS" for the Primary Data Source and "SecondaryDS<#>" for the corresponding Secondary Data Source. For example: "PrimaryDS.Database=c:\DBs\db1.mdb;SecondaryDS1.Database=c:\DBs\db2.mdb"

Remarks: The parameters used to create the Database are set at form design time using the "Database Relations" facility of the Designer.

Return Values: True - Call to this method was Successful
False - An Error was encountered

See Also: GetLastError(), Connect()

Delete

Description::	Delete the current Record from the database.
Syntax:	<div>C++ Visual Basic</div> <div>BOOL CMmaADOi::Delete(); [form.] MmaADOi.Delete()</div>
Parameters:	None
Remarks:	Use this method if you do not want to use the ActiveX's own User Interface for Deleting the current record from the database.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
See Also:	“GetLastError” on page 344

Disconnect

Description:	Closes connection to the database.
Syntax:	<div>C++ Visual Basic</div> <div>BOOL CMmaADOi::Disconnect(); [form.] MmaADOi.Disconnect()</div>
Parameters:	None
Remarks:	To re-establish connection between the form and the database, you need to invoke the Connect() method.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
See Also:	GetLastError(), Connect()

FindFirst | FindLast | FindNext | FindPrevious

Description:	Locate the first, last, next, or previous record in the database that satisfies the specified criteria and make that record the current record.
Syntax:	<div>C++<pre>BOOL CMmaADOi:: {FindFirst FindLast FindNext FindPrevious} (LPCTSTR Criteria);</pre></div> <div>Visual Basic<pre>[form.] MmaADOi. {FindFirst FindLast FindNext FindPrevious} (Criteria As String)</pre></div>
Parameters:	Criteria A string expression (the WHERE clause in an SQL statement without the word WHERE) used as the Find condition.
Remarks:	<p>Use the Find methods to locate records that satisfy a specific condition. If you want to include all the records in your search - not just those that meet a specific condition - use the Move methods to move from record to record.</p> <p>If table contains more than one record that satisfies criteria, FindFirst locates the first occurrence, FindNext locates the next one, and so on. Using one of the Find methods isn't the same as using MoveFirst or MoveNext, however, which simply makes the first or next record current without applying a condition. You can follow a Find operation with a Move operation.</p>
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
See Also:	GetLastError(), MoveFirst(), MoveLast(), MoveNext(), MovePrevious()

GetAbsolutePosition

Description:	Returns the relative record number of the current record.
Syntax:	<div>C++ LONG CMmaADOi::GetAbsolutePosition);</div> <div>Visual Basic [form.] MmaADOi.GetAbsolutePosition ()</div>
Parameters:	None
Remarks:	<p>You can determine the current record number by checking the GetAbsolutePosition method setting.</p> <p>You can determine the number of populated records in the table by using the GetRecordCount method.</p> <p>If there is no current record, as when there are no records in the table, GetAbsolutePosition returns -1. If the current record is deleted, the GetAbsolutePosition method value isn't defined. New records are added to the end of the sequence.</p>
Return Values:	The return value is a Long integer from 0 to one less than the number of records in the table. It corresponds to the ordinal position of the current record in the table.
See Also:	GetLastError(), GetRecordCount()

GetLastError

Description:	Returns the message corresponding to the most recent run-time error generated by a call to one of MmaADOi methods.
Syntax:	<div>C++ Visual Basic</div> <div>VARIANT CMmaADOi::GetLastError(); [form.] MmaADOi.GetLastError ()</div>
Parameters:	None
Remarks:	None
Return Values:	Message corresponding to the most recent run-time error
See Also:	None

GetRecordCount

Description:	Returns the total number of records in a table.
Syntax:	<div>C++ LONG CMmaADOi::GetRecordCount();</div> <div>Visual Basic [form.] MmaADOi.GetRecordCount ()</div>
Parameters:	None
Remarks:	A table with no records has a Record Count of 0.
Return Values:	The return value is a Long data type.
See Also:	“GetLastError” on page 344

Lookup

Description: Displays a Lookup Dialog containing a selectable list of all records matching a Search Criteria.

Syntax:

C++	VARIANT CMmaADOi::Lookup(SHORT DataSourceIndex, LPCTSTR FieldName, LPCTSTR SearchCriteria, LPCTSTR DialogCaption);
Visual Basic	[form.] MmaADOi.Lookup(DataSourceIndex As Integer, FieldName As String, SearchCriteria As String, DialogCaption as String)

Parameters: The following parameters are available

Parameter	Description
DataSourceIndex	Use 0 for Primary Data Source; 1 for Secondary Data Source 1, etc.
FieldName	This Field is updated with the user's selection
SearchCriteria	Criteria by which a list of records are displayed.
DialogCaption	Caption of the Lookup Dialog

Remarks: If the user cancels out of the Lookup Dialog, 'FieldName' will not be updated.

Return Values:

True - Call to this method was Successful

False - An Error was encountered

See Also: "GetLastError" on page 344

MoveFirst | MoveLast | MoveNext | MovePrevious

Description:	Move to the first, last, next, or previous record in a specified table and make that record the current record.
Syntax:	<div>C++<pre>BOOL CMmaADOi:: {MoveFirst MoveLast MoveNext movePrevious} ();</pre></div> <div>Visual Basic<pre>[form.] MmaADOi. {MoveFirst MoveLast MoveNext movePrevious} ()</pre></div>
Parameters:	None
Remarks:	Use the Move methods to move from record to record without applying a condition.
Caution:	If you edit the current record, be sure you use the Update method to save the changes before you move to another record. If you move to another record without updating, your changes are lost without warning.
Return Values:	<div>True - Call to this method was Successful</div> <div>False - An Error was encountered</div>
See Also:	“GetLastError” on page 344

Update

Description: Saves the contents of the copy buffer to the table.

Syntax: C++ `BOOL CMmaADOi::Update();`
Visual Basic `[form.] MmaADOi.Update()`

Parameters: None

Remarks: Use Update to save the current record and any changes you've made to it.

Caution: Changes to the current record are lost if:

- You use the AddNew method, and then move to another record without first using Update.
- You use AddNew, and then use AddNew again without first using Update.

Return Values: True - Call to this method was Successful
False - An Error was encountered.

See Also: "GetLastError" on page 344

Events

RecordAdd

Description:	Occurs when the AddNew Method is invoked.
Syntax:	<div>C++ afx_msg void RecordAddMmaADOi ()</div> <div>Visual Basic Sub MmaADOi_RecordAdd()</div>
Parameters:	None.
Remarks:	None.

RecordDelete

Description:	Occurs when the Delete Method is completed.
Syntax:	<div>C++ afx_msg void RecordDeleteMmaADOi ()</div> <div>Visual Basic Sub MmaADOi_RecordDelete()</div>
Parameters:	None.
Remarks:	None.

RecordMove

Description:	Occurs when any of the Find or Move methods is invoked causing the current record to change.
Syntax:	<div>C++ afx_msg void RecordMoveMmaADOi ()</div> <div>Visual Basic Sub MmaADOi_RecordMove()</div>
Parameters:	None.
Remarks:	None.

RecordUpdate

Description:	Occurs when the Update Method is completed.
Syntax:	C++ afx_msg void RecordUpdateMmaADOi () Visual Basic Sub MmaADOi_RecordUpdate()
Parameters:	None.
Remarks:	None.

Index

Symbols

. 28

A

Accessibility 90
ActiveX 147
Adding pages 22
Aligning objects 37
Appearance properties 43
Assigning database relations 154
Auto tab 52

B

Background color, objects 44
Bar code object button 8
Bar codes 73
Borders 45
Bottom align button 10
Box object button 7
Breaking a group of objects 38
Built-in functions 98
Button object button 7
Buttons 64
 changing properties 64

C

Calculations 54
 details 96
Changing the properties of objects 39
Character Spacing 50
Check box object button 7
Check boxes 62
Circle object button 7
Coding 147
Conventions in this manual 2
Converting objects 38
Copy button 9
Creating forms 16
Crosshairs 31

Currency object button 7
Cut button 9

D

Database connectivity 147
Database format 149
Database relations 150
Database toolbar 151
Date object button 7
Default value, creating 55
Defining form setup 17
Deleting objects 38
Digital signature object button 8
Digital signatures 75, 77
 creating 75
 drawing 77
Drawing a fillable object 59
Drawing an object 30
Drop down object button 8
Drop lists 67

E

Edit field object button 7
Edit object button 7
Edit properties 52
Editable image object button 8
Editable images 66
Editing text 83
Editing text in an object, selecting 83
Editing text in forms 83
Entering text in a form 83
Export 26
Expressions 101

F

Fill button 9
Fill character, creating 55
Fill font 50
 formatting 50

Fill mode 87
Fillable objects 29
Filling forms 87
Finding and replacing text 85
Font format button 10
Fonts
 resizing 50
 selecting fonts 50
 style 50
Form design guide 14
Form Properties 24
Form setup 17
Formatting tables 69
FormFlow 26
Functions 96

G

Grid, displaying 34
Grouping objects 38

H

Help and support 2
Help button 9
Horizontal center align button 10
Hour function 113

I

If function 121
Image object button 8
Images 65
 defining properties 65
Import 26
Installation and setup 3

L

Layering overlapping objects 37
Layout. See also Form setup
Left align button 10
Left function 122, 123, 124, 129, 131
Line Thickness 43
Lower function 123
Ltrim function 124

M

Mandatory, changing 52
Mandatory, enabling and disabling 52
Margin properties 47
Margin width, determining 47
Mask object button 7
Mask, example of a social security 60
Masks 59
Menus 6
Miscellaneous properties, changing
 name, nonprintable 40
MmaADOi 147
Modifying objects 38
Move an object 38
Multiline, enabling and disabling 52

N

New form button 9
New forms 16
Next page button 10
Non-fillable objects 28
Notify flags 57
Notify properties 57
Num function 127
Number object button 7

O

Object buttons explained 7
Object toolbar 6
Objects 28
Opaque, making an object 44
Open form button 9
Operators 102
Overwrite, enabling and disabling 53

P

Page Setup 18
Paste button 9
Position properties 42
Positioning objects 32
Positioning objects by coordinates 42
Previous page button 10
Primary and secondary data source 151
Print button 9

Printing forms 86
Properties window 11
Properties window, using 39
Property control bar 12, 80

R

Redo button 9
Resize an object 38
Right align button 10
Right function 129
Round function 130
Rtrim function 131

S

Same height button 10
Same height, making objects 37
Same width button 10
Same width, making objects 37
Save form button 9
Saving forms 20
Screen Components 6
Scripts 94
Select object button 7
Selecting multiple objects 32
Selecting objects 32
Selecting special objects 32
Selecting tables 32
Set property function 135
Setup 3
Sizing objects 43
Snap to grid, using 36
Special properties 59
Spell check button 11
Spell Checking 84
Standard toolbar 9
Straight line object button 7
Strat function 137, 138
Strextract function 138
Strinstr function 139

Strlen function 140
Sum function 141
System requirements 3

T

Tab order of objects, changing 53
Table object button 8
Tables 69
Tables, defining the properties in 69
Text characters, maximum 52
Text font 50
 formatting 50
 resizing 50
Text justification, selecting 49
Text object button 7
Text orientation, selecting 48
Text properties 48
Text spacing, selecting 48
Time function 142, 143, 144
Top align button 10

U

Undo button 9
Upper function 145, 146
Using database relations 158

V

Vertical center align button 10
Viewing forms 21
Visible outline 45
Visual eForms Designer
 Installing 3
 Starting 5

Z

Zoom 21
Zoom page button 10